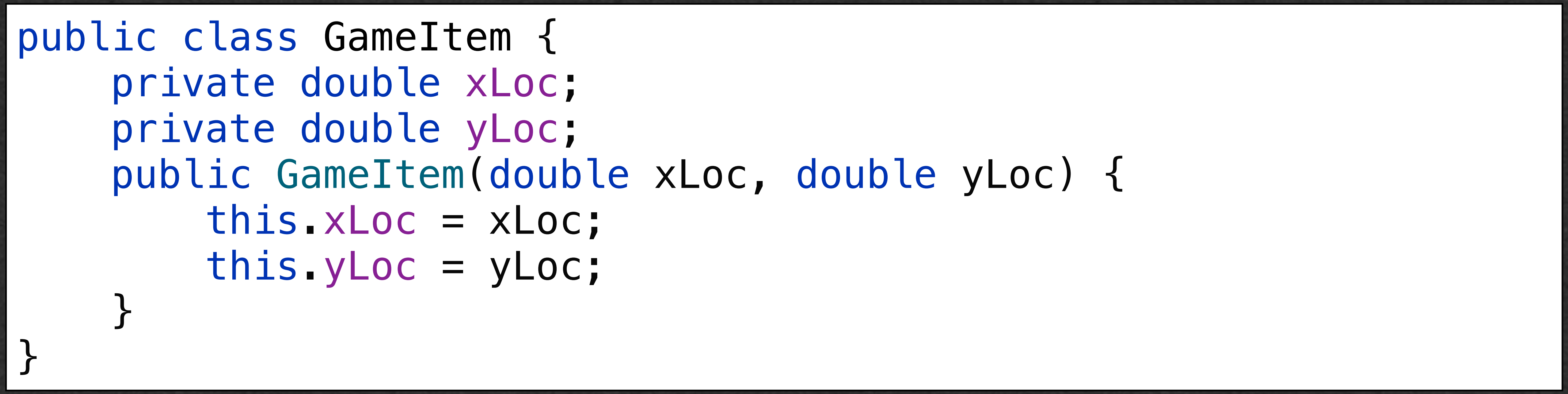
# Inheritance

# Override

# Inheritance

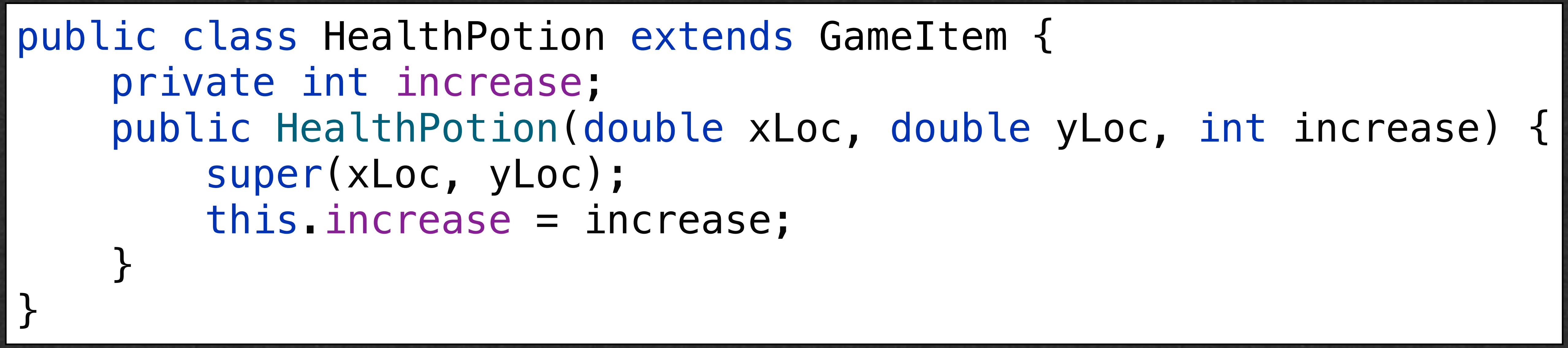
- Recall that we used inheritance to add all of the state and behavior of one class to another class



```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
}
```

- HealthPotion extends (or, inherits from) GameItem
  - HealthPotion objects have all the instance variables (State) of both HealthPotion and GameItem

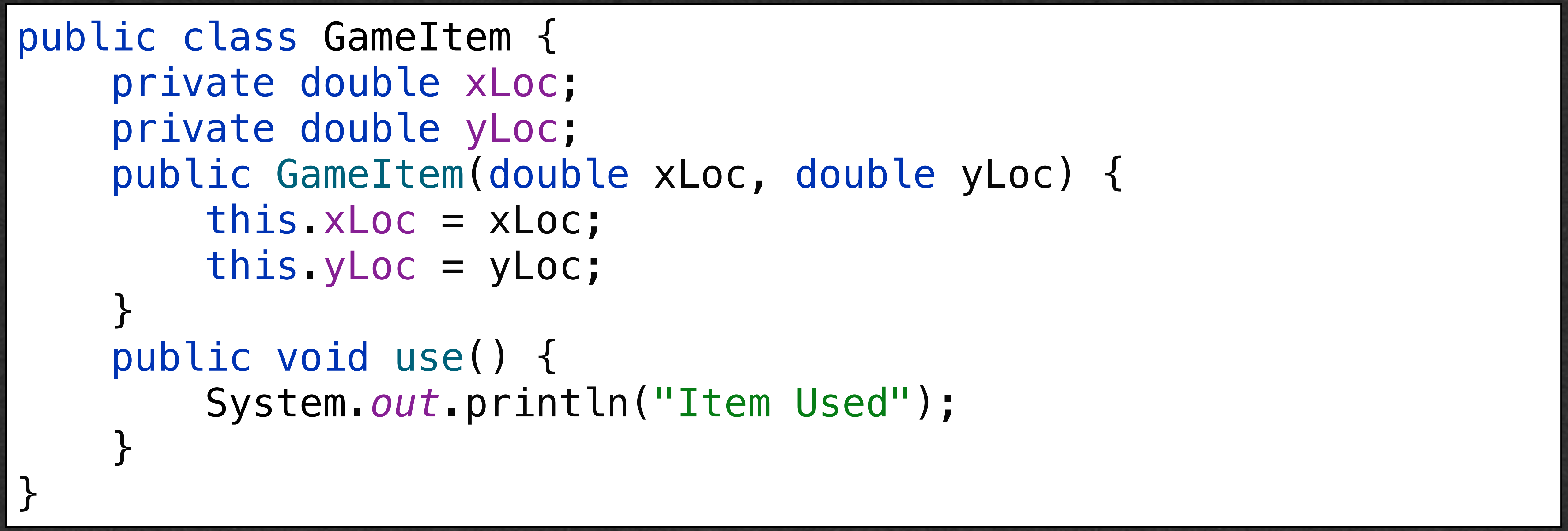
- GameItem is the super class of HealthPotion



```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
}
```

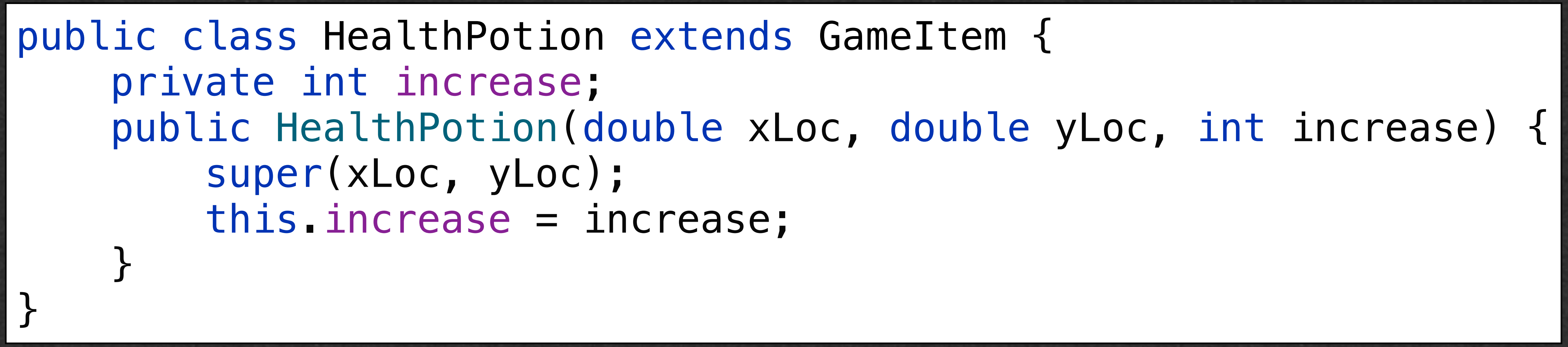
# Inheritance

- HealthPotion objects have all the methods (Behavior) of both HealthPotion and GameItem



```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    public void use() {
        System.out.println("Item Used");
    }
}
```

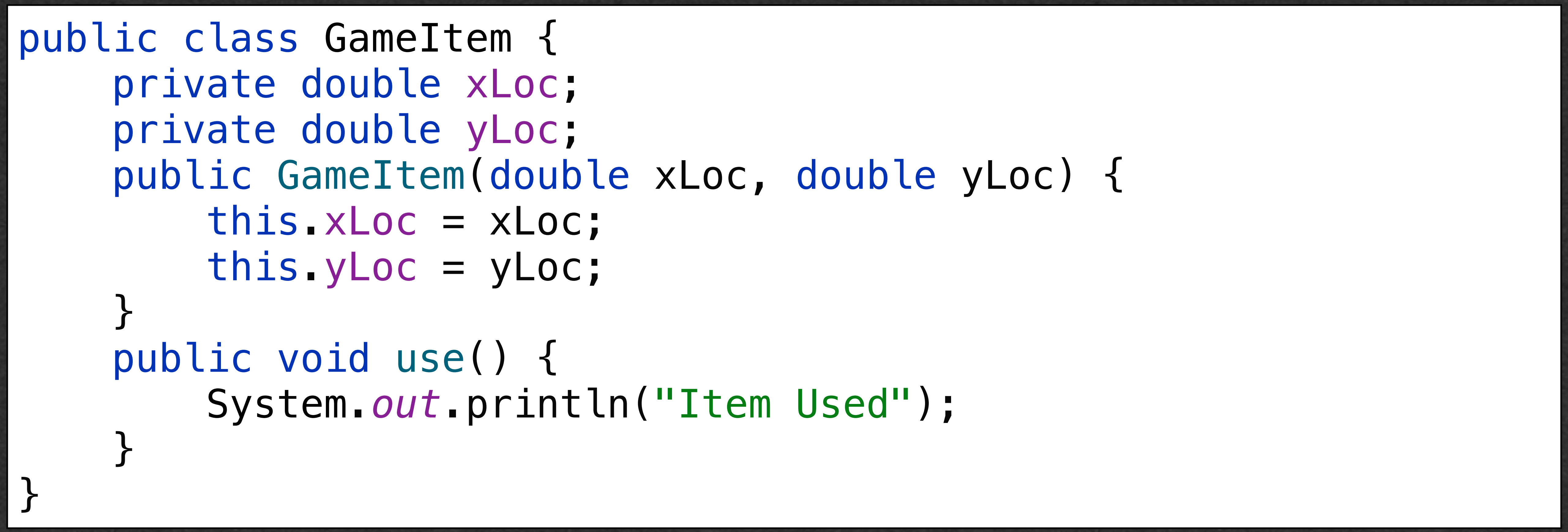
- We add a use method to the HameItem class
  - All HealthPotion objects now have a use method



```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
}
```

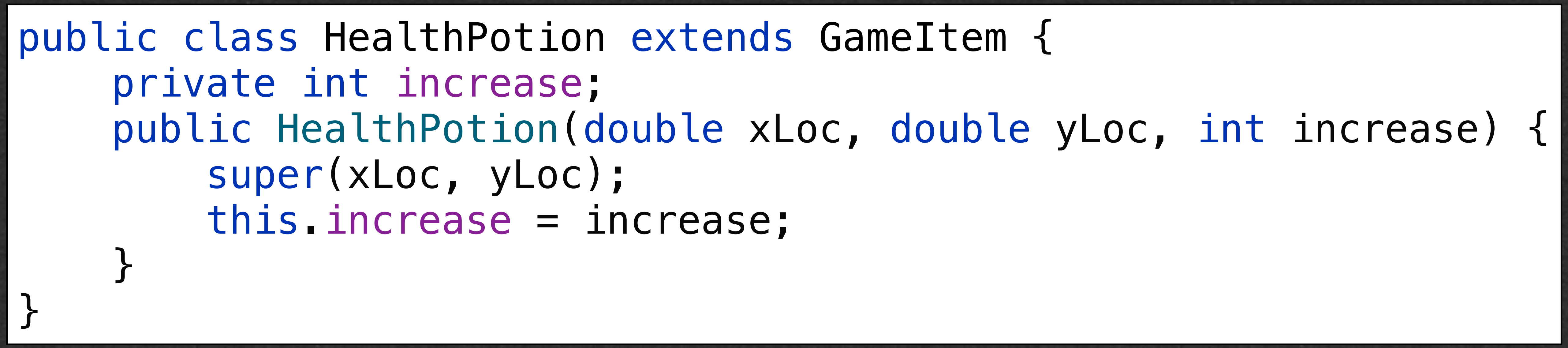
# Inheritance

- What if we want to extend a class, but don't want 100% of the inherited state and behavior?



```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    public void use() {
        System.out.println("Item Used");
    }
}
```

- We want a class to inherit the location code from GameItem, but want the use method to something else
    - Override!

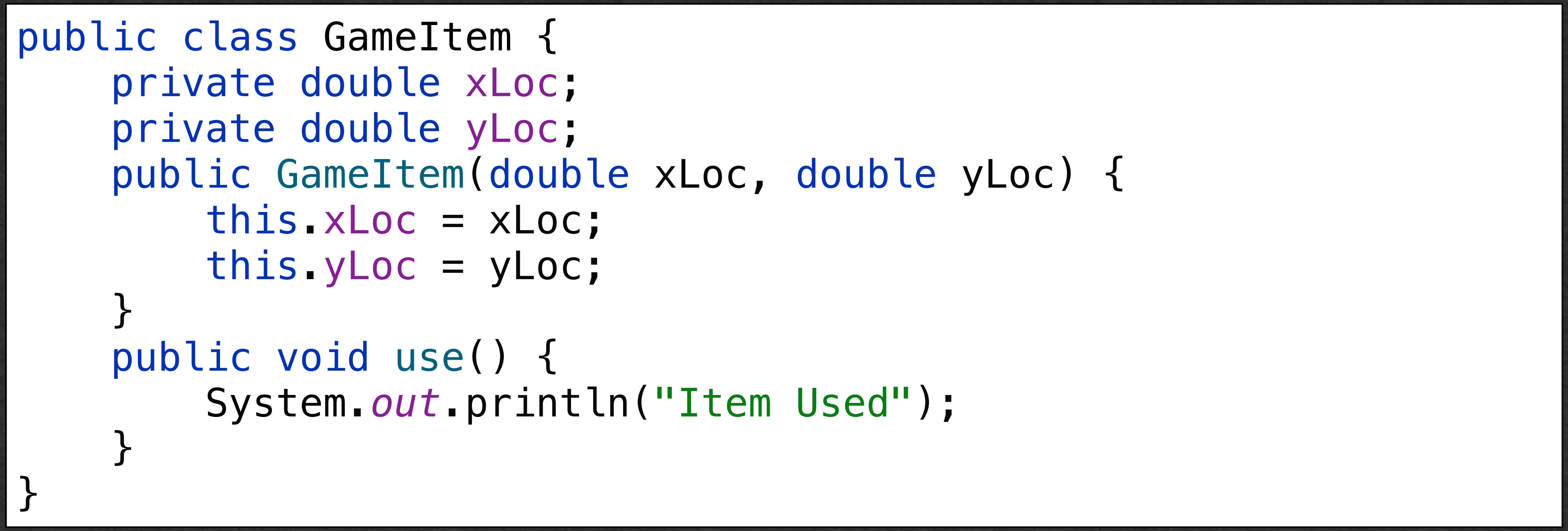


```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
}
```

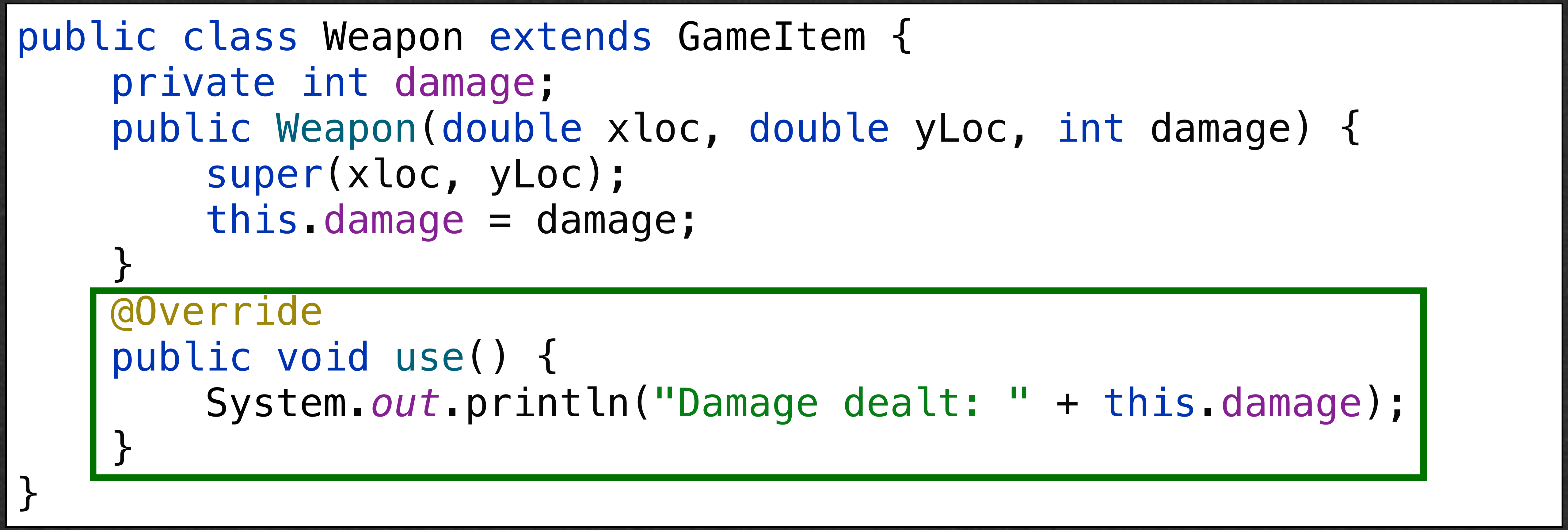
# Override

- Weapon will also inherit the state and behavior from GameItem

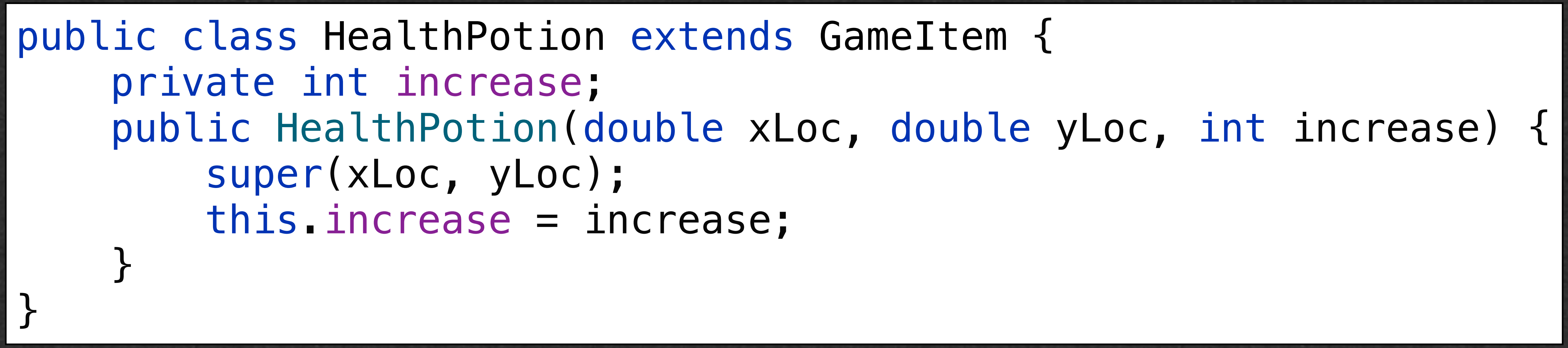
- We will **Override** the use method with a new definition specific to the Weapon class
    - The inherited method is **replaced** by this new definition



```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    public void use() {
        System.out.println("Item Used");
    }
}
```



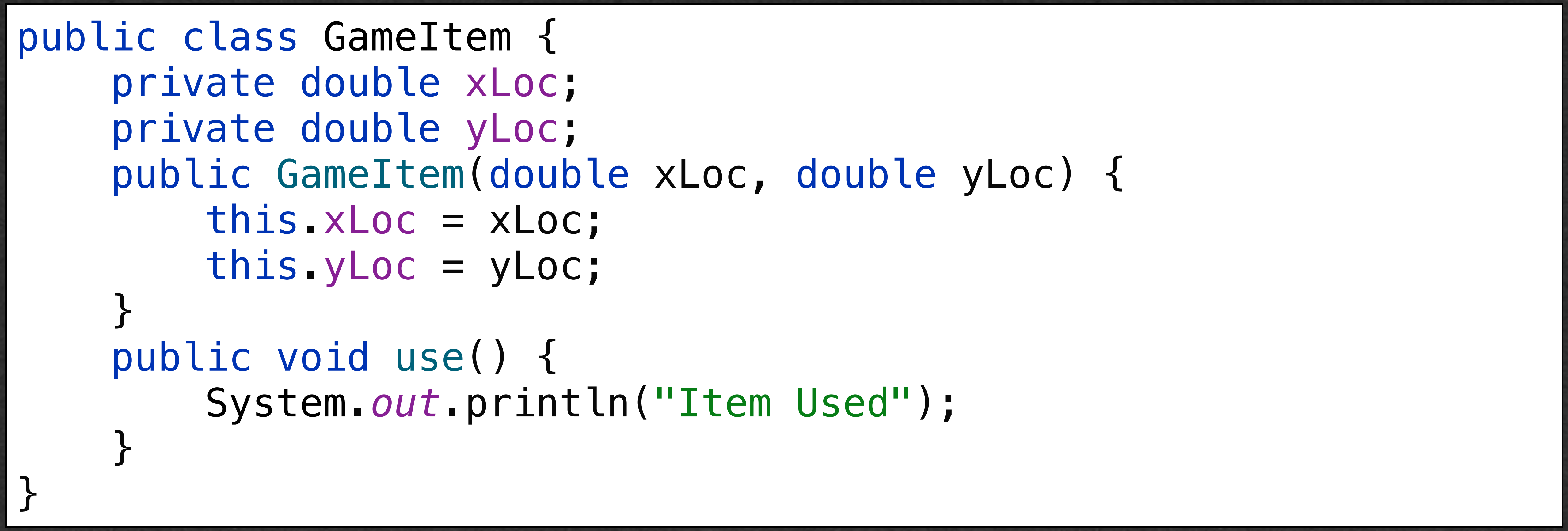
```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    @Override
    public void use() {
        System.out.println("Damage dealt: " + this.damage);
    }
}
```



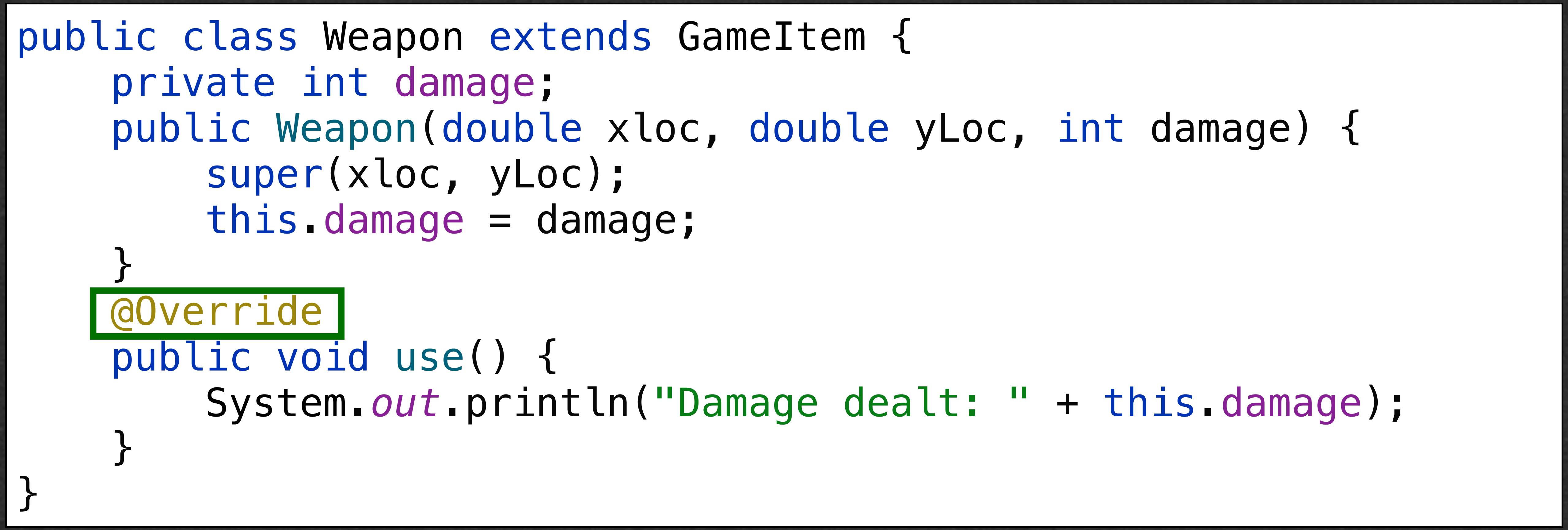
```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
}
```

# Override

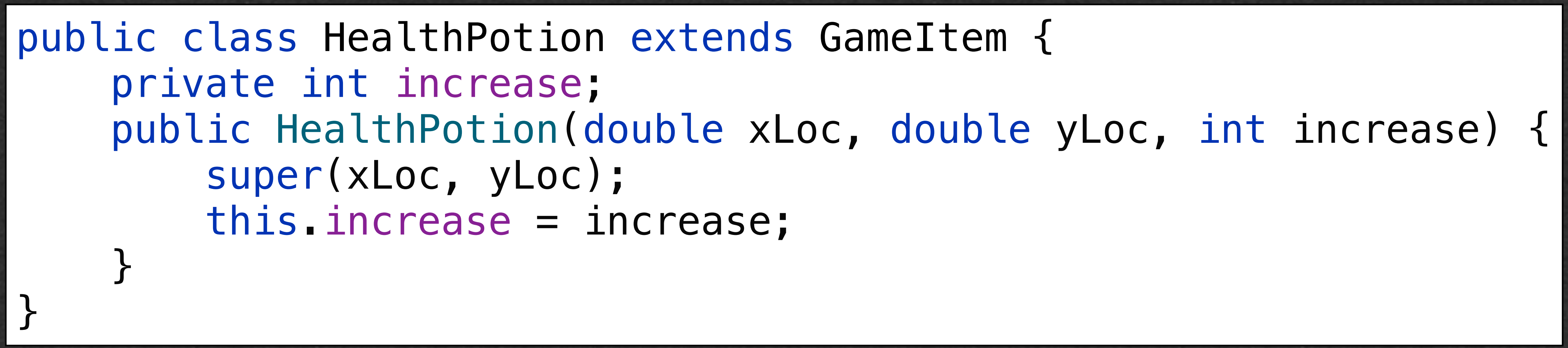
- To Override a method definition
  - Use the @Override annotation before the method
  - The annotation makes your intentions clear and tells the compiler that this method will replace an inherited method



```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    public void use() {
        System.out.println("Item Used");
    }
}
```



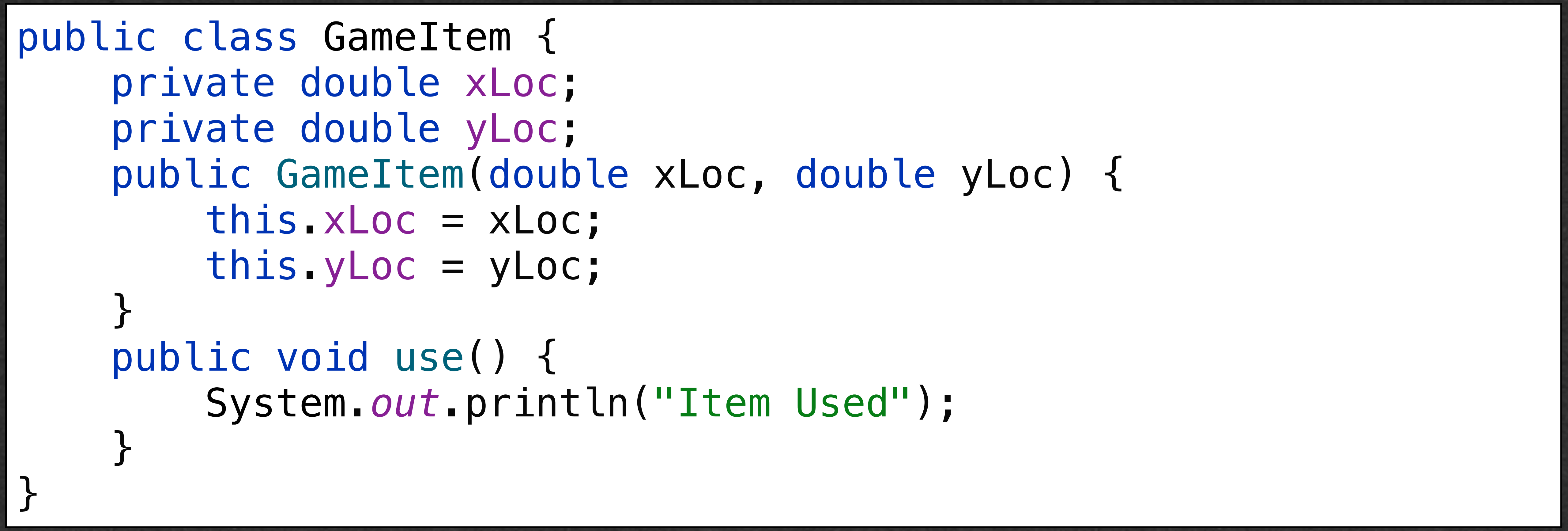
```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    @Override
    public void use() {
        System.out.println("Damage dealt: " + this.damage);
    }
}
```



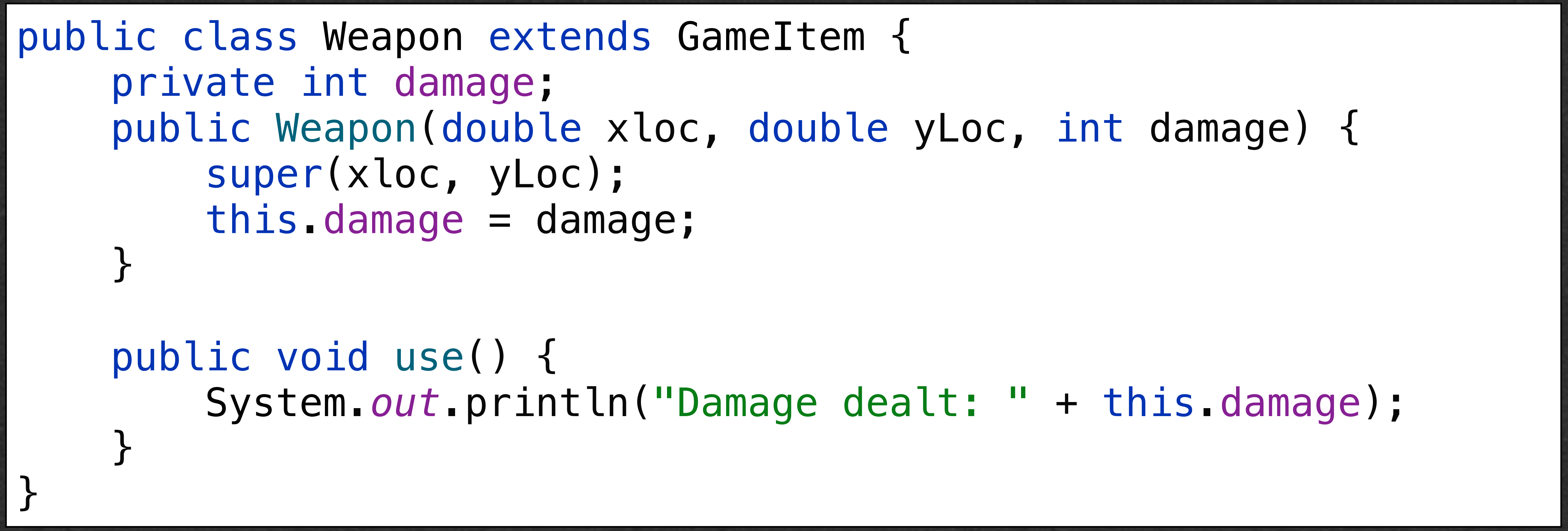
```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
}
```

# Override

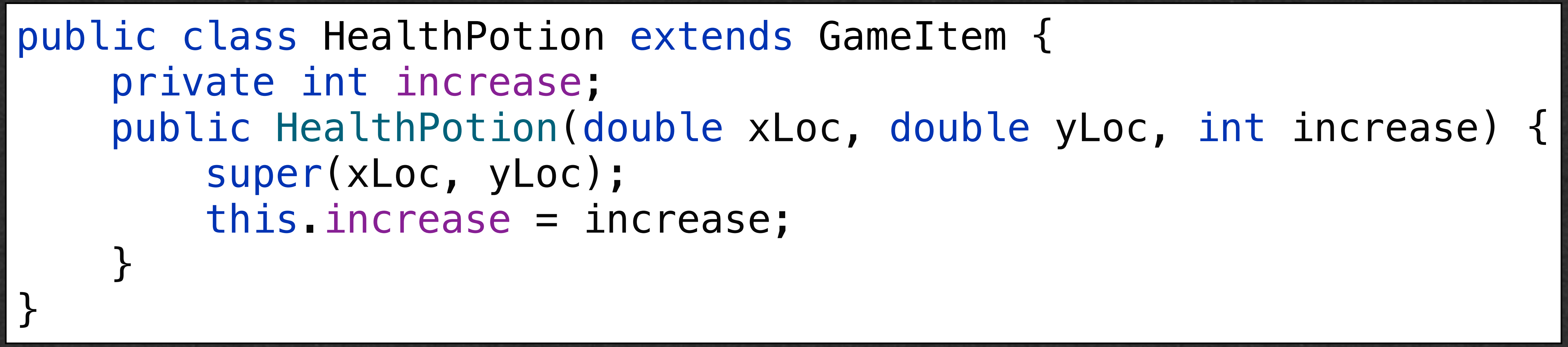
- The @Override annotation is optional [but recommended]
- When overriding a method, your method must have the same *signature* as the method being overwritten
  - Same name
  - Same number of parameters
  - Same parameter types
  - Same return type
- If there are any differences between the methods, the method is not overridden



```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    public void use() {
        System.out.println("Item Used");
    }
}
```



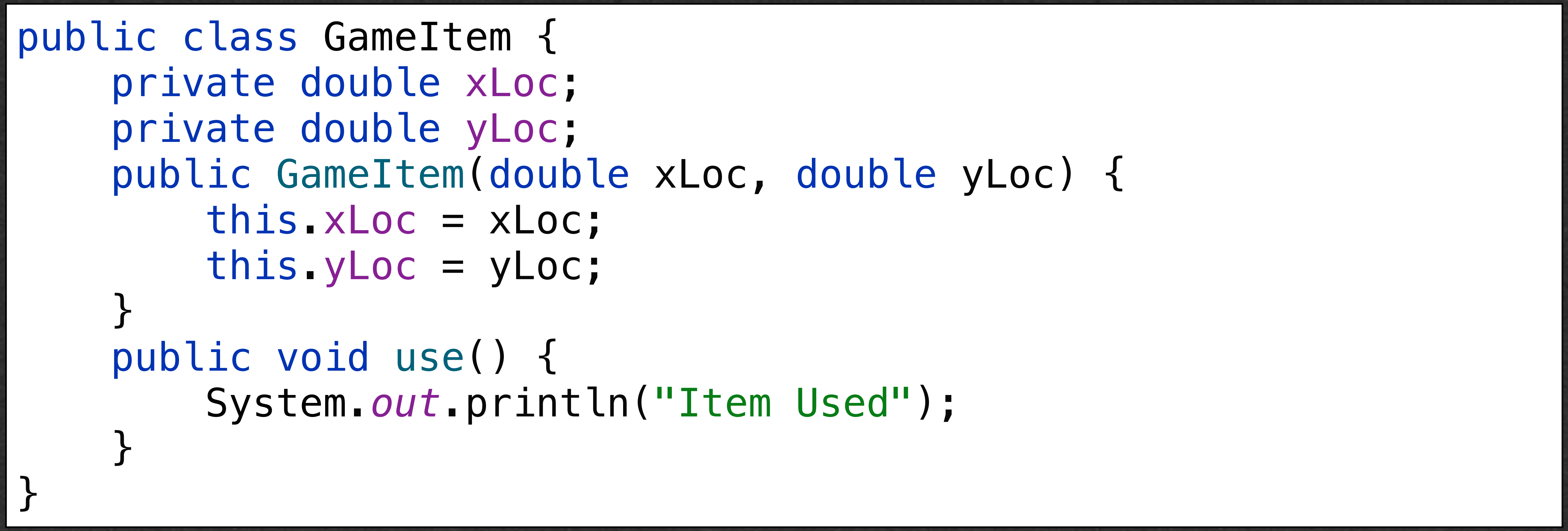
```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }

    public void use() {
        System.out.println("Damage dealt: " + this.damage);
    }
}
```



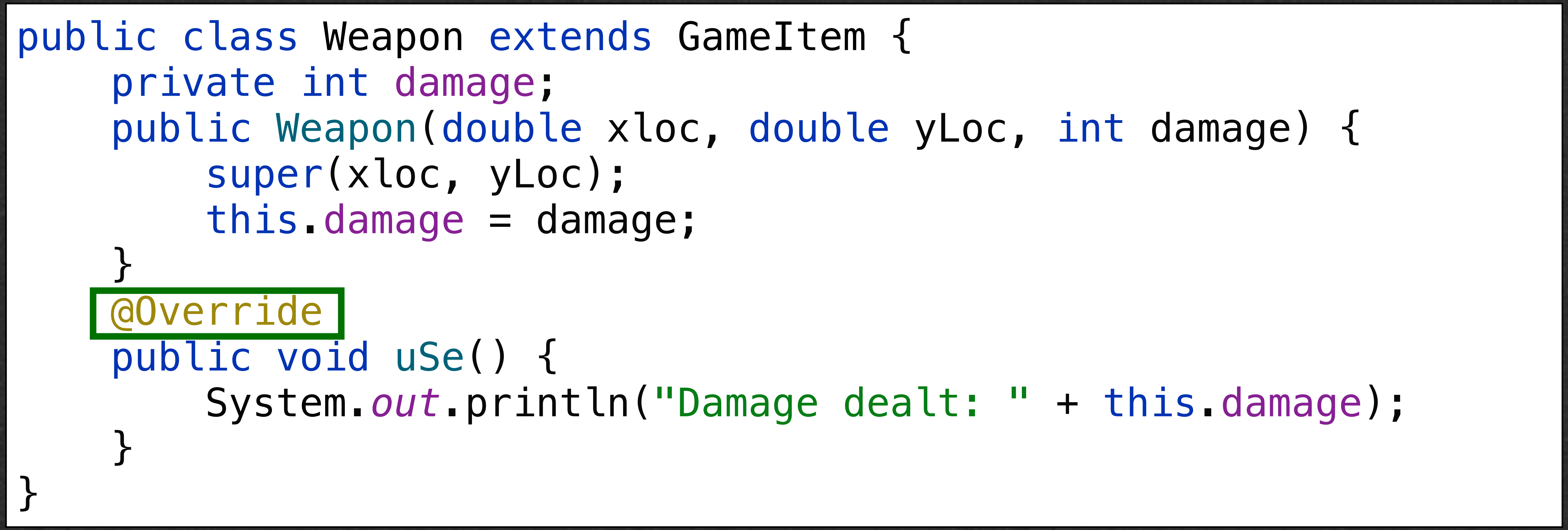
```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
}
```

# Override

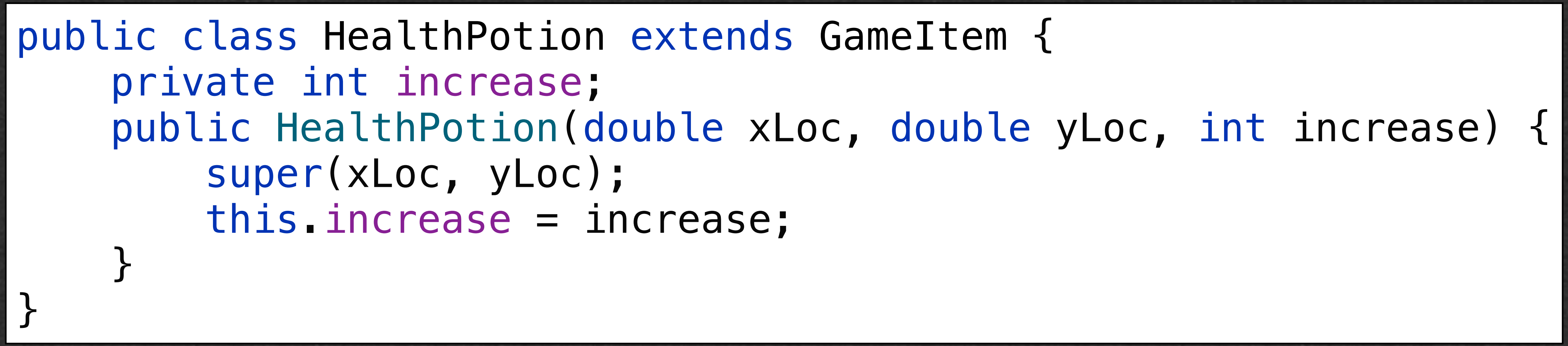
- If you have the @Override annotation
  - The compiler will let you know if you have mistakes in the method signature
- This code will not compile since uSe does not match the signature of any inherited method
- Without the @Override annotation:
  - This code will compile and run, but will not do what you want or expect



```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    public void use() {
        System.out.println("Item Used");
    }
}
```



```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    @Override
    public void uSe() {
        System.out.println("Damage dealt: " + this.damage);
    }
}
```



```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
}
```

# Incoming Memory Diagram!!

```
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    public void use() {
        System.out.println("Item Used");
    }
}
```

```
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    @Override
    public void use() {
        System.out.println("Damage dealt: " + this.damage);
    }
}
```

```
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
}
```

```
public static void main(String[] args) {
    Weapon weapon = new Weapon(1.3, 0.7, 100);
    HealthPotion potion = new HealthPotion(10.0, 0.0, 6);
    weapon.use();
    potion.use();
}
```

| Stack | | Heap |
|---|---|---|
| Name | Value | |
| | | in/out |

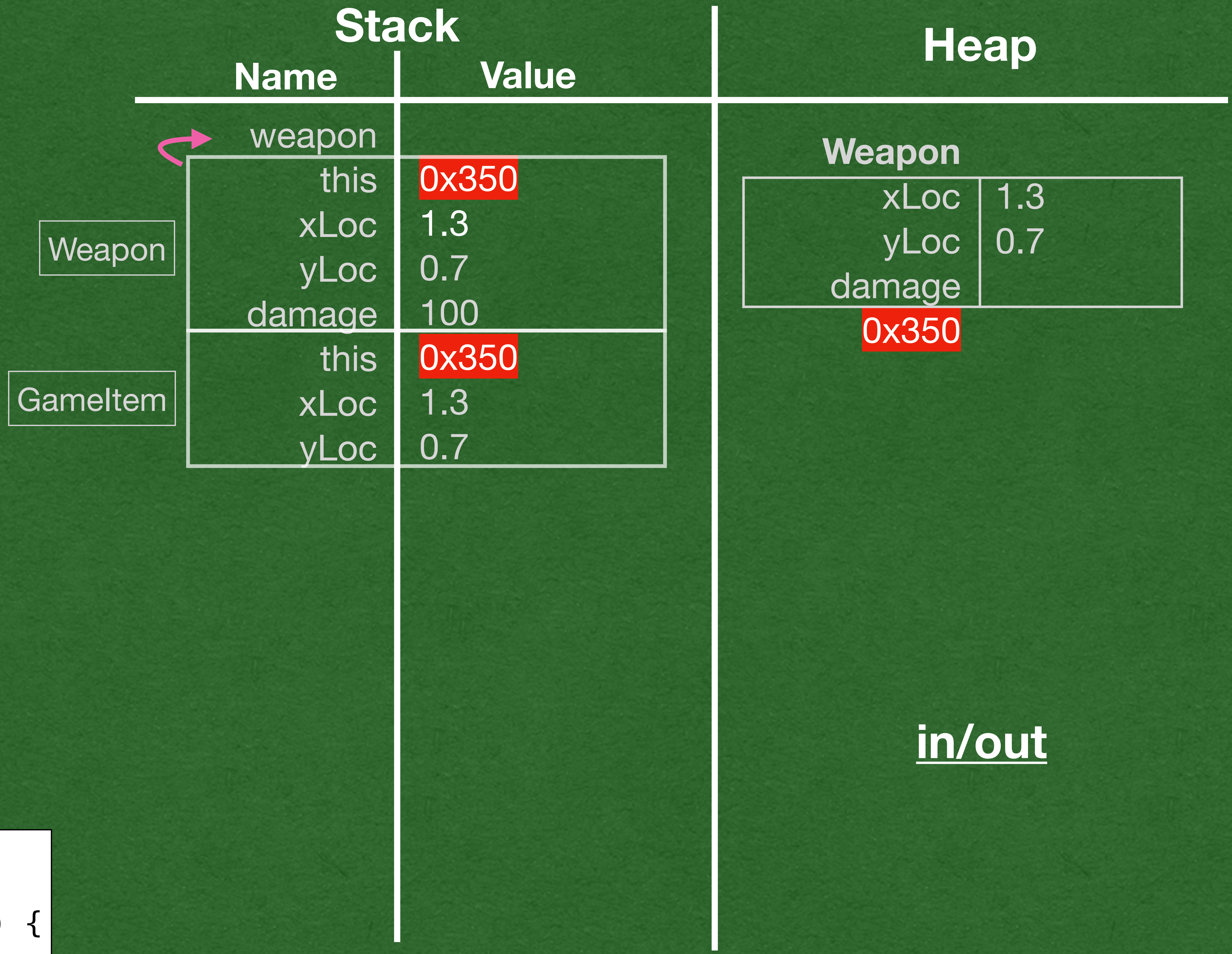
- What will happen when the use method is called?
- There are 2 definitions of the method in 2 different classes

```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    public void use() {
        System.out.println("Item Used");
    }
}
```

```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    @Override
    public void use() {
        System.out.println("Damage dealt: " + this.damage);
    }
}
```

```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
}
```

```java
public static void main(String[] args) {
    Weapon weapon = new Weapon(1.3, 0.7, 100);
    HealthPotion potion = new HealthPotion(10.0, 0.0, 6);
    weapon.use();
    potion.use();
}
```



## Stack

| | Name | Value |
|---|---|---|
| | weapon | |
| Weapon | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | damage | 100 |
| GameItem | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |

## Heap

**Weapon**

| | |
|---|---|
| xLoc | 1.3 |
| yLoc | 0.7 |
| damage | |

0x350

**in/out**

When a class extends another class:

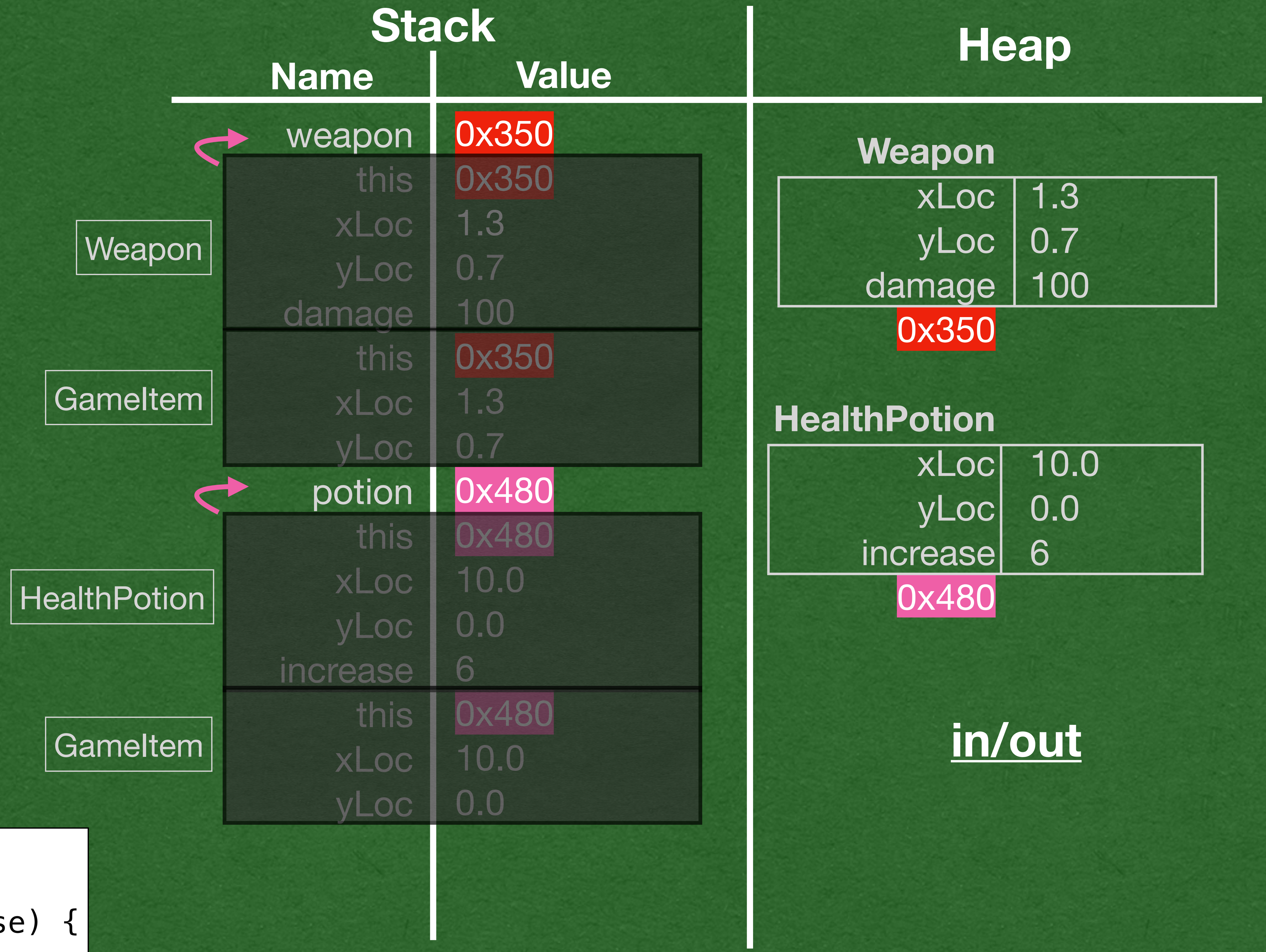
- Objects inherit all the instance variables of the super class
- The super class constructor is called (Do not forget this stack frame)

```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    public void use() {
        System.out.println("Item Used");
    }
}
```

```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    @Override
    public void use() {
        System.out.println("Damage dealt: " + this.damage);
    }
}
```

```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
}
```

```java
public static void main(String[] args) {
    Weapon weapon = new Weapon(1.3, 0.7, 100);
    HealthPotion potion = new HealthPotion(10.0, 0.0, 6);
    weapon.use();
    potion.use();
}
```



## Stack

| | Name | Value |
|---|---|---|
| | weapon | 0x350 |
| Weapon | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | damage | 100 |
| GameItem | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | potion | 0x480 |
| HealthPotion | this | 0x480 |
| | xLoc | 10.0 |
| | yLoc | 0.0 |
| | increase | 6 |
| GameItem | this | 0x480 |
| | xLoc | 10.0 |
| | yLoc | 0.0 |

## Heap

**Weapon**

| | |
|---|---|
| xLoc | 1.3 |
| yLoc | 0.7 |
| damage | 100 |

0x350

**HealthPotion**

| | |
|---|---|
| xLoc | 10.0 |
| yLoc | 0.0 |
| increase | 6 |

0x480

**in/out**

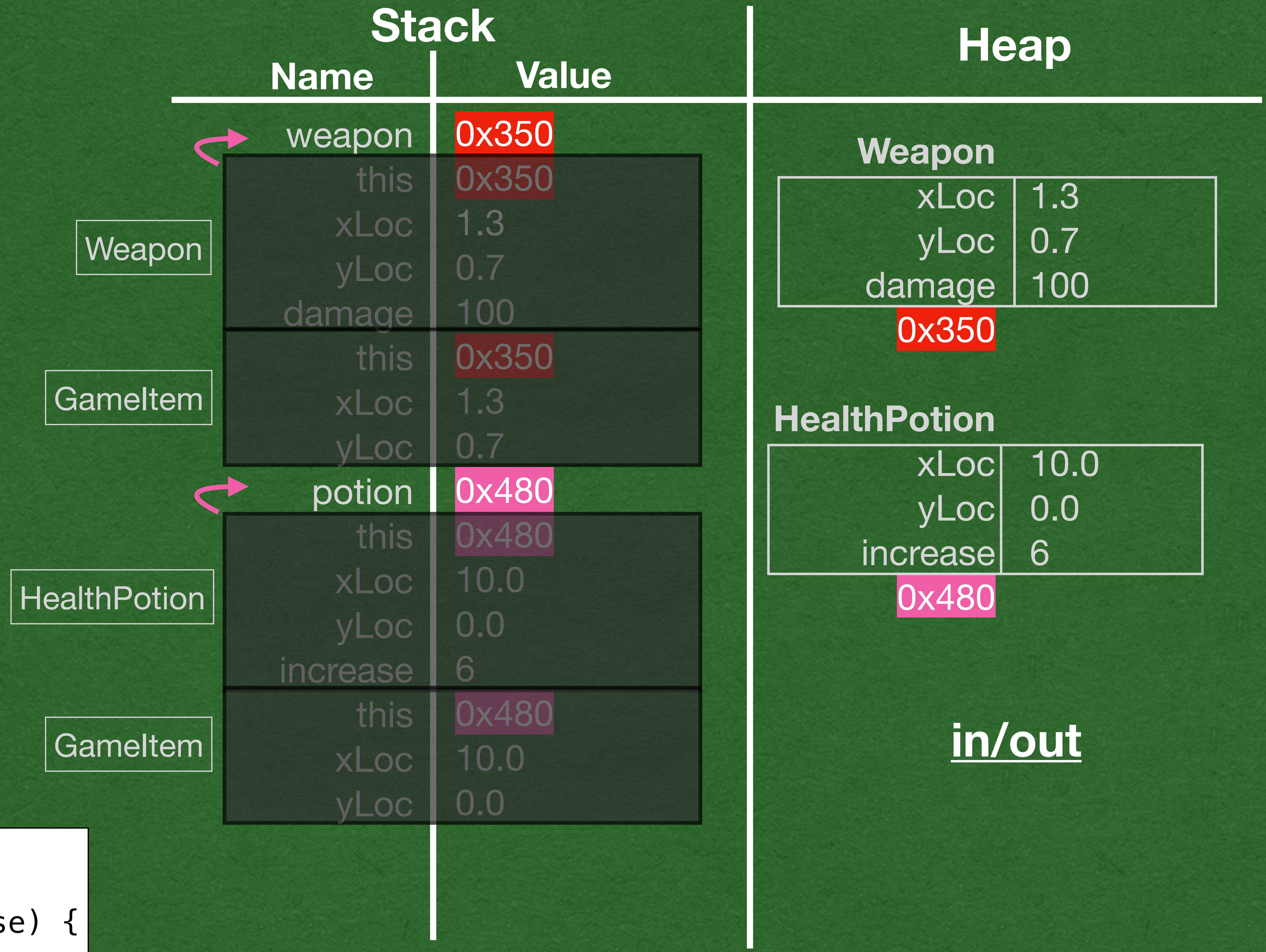
- The same applies to HealthPotion
- Do not forget the super constructor stack frame

```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    public void use() {
        System.out.println("Item Used");
    }
}
```

```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    @Override
    public void use() {
        System.out.println("Damage dealt: " + this.damage);
    }
}
```

```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
}
```

```java
public static void main(String[] args) {
    Weapon weapon = new Weapon(1.3, 0.7, 100);
    HealthPotion potion = new HealthPotion(10.0, 0.0, 6);
    weapon.use();
    potion.use();
}
```



**Stack**

|  | Name | Value |
|---|---|---|
|  | weapon | 0x350 |
| Weapon | this | 0x350 |
|  | xLoc | 1.3 |
|  | yLoc | 0.7 |
|  | damage | 100 |
| GameItem | this | 0x350 |
|  | xLoc | 1.3 |
|  | yLoc | 0.7 |
|  | potion | 0x480 |
| HealthPotion | this | 0x480 |
|  | xLoc | 10.0 |
|  | yLoc | 0.0 |
|  | increase | 6 |
| GameItem | this | 0x480 |
|  | xLoc | 10.0 |
|  | yLoc | 0.0 |

**Heap**

**Weapon**

|  |  |
|---|---|
| xLoc | 1.3 |
| yLoc | 0.7 |
| damage | 100 |

0x350

**HealthPotion**

|  |  |
|---|---|
| xLoc | 10.0 |
| yLoc | 0.0 |
| increase | 6 |

0x480

**in/out**

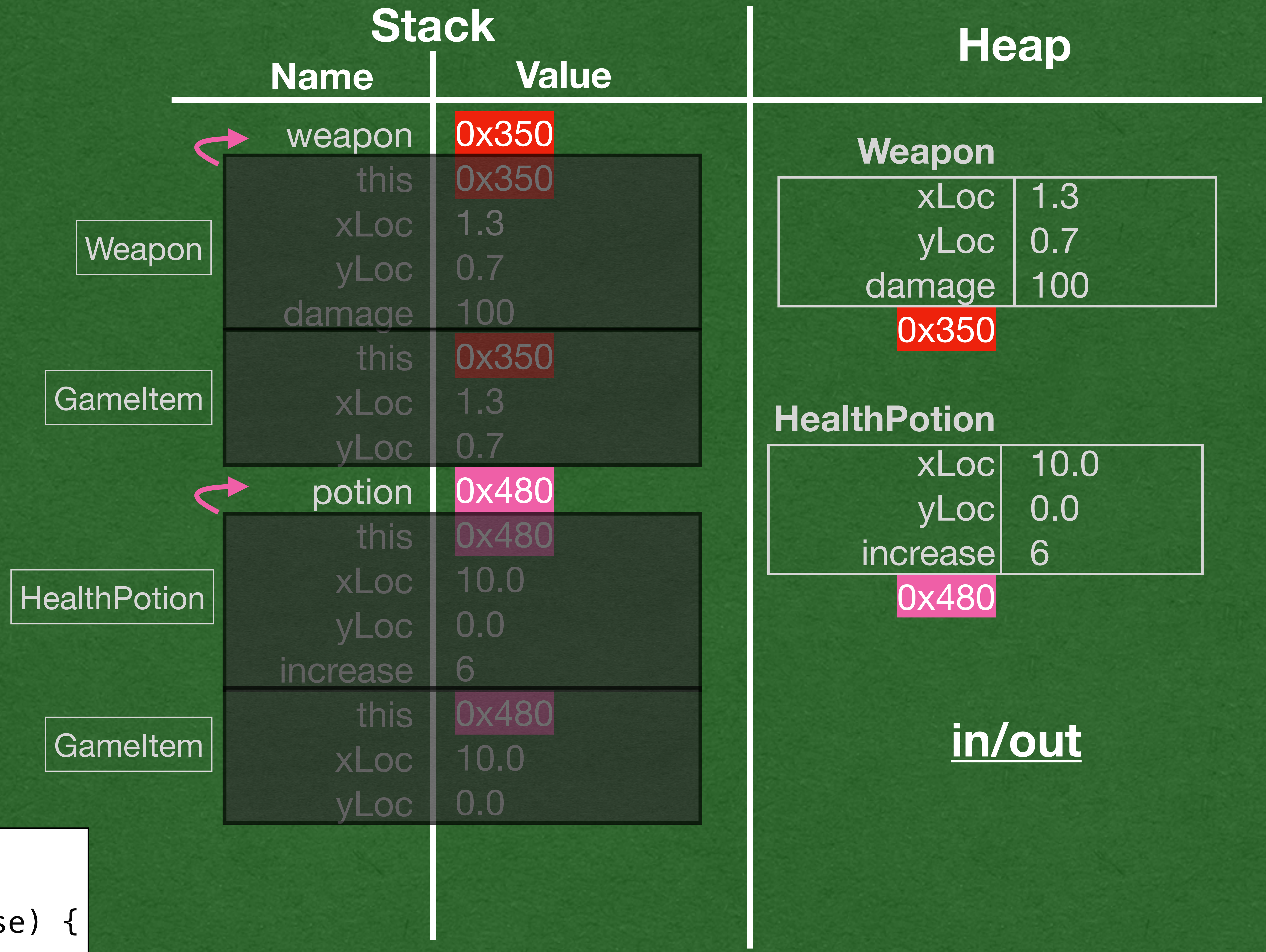
- We are calling the use method
- What method will be called?
    - There are 2 different use methods

```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    public void use() {
        System.out.println("Item Used");
    }
}
```

```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    @Override
    public void use() {
        System.out.println("Damage dealt: " + this.damage);
    }
}
```

```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
}
```

```java
public static void main(String[] args) {
    Weapon weapon = new Weapon(1.3, 0.7, 100);
    HealthPotion potion = new HealthPotion(10.0, 0.0, 6);
    weapon.use();
    potion.use();
}
```



## Stack

| | Name | Value |
|---|---|---|
| | weapon | 0x350 |
| Weapon | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | damage | 100 |
| GameItem | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | potion | 0x480 |
| HealthPotion | this | 0x480 |
| | xLoc | 10.0 |
| | yLoc | 0.0 |
| | increase | 6 |
| GameItem | this | 0x480 |
| | xLoc | 10.0 |
| | yLoc | 0.0 |

## Heap

Weapon

| | |
|---|---|
| xLoc | 1.3 |
| yLoc | 0.7 |
| damage | 100 |

0x350

HealthPotion

| | |
|---|---|
| xLoc | 10.0 |
| yLoc | 0.0 |
| increase | 6 |

0x480

**in/out**

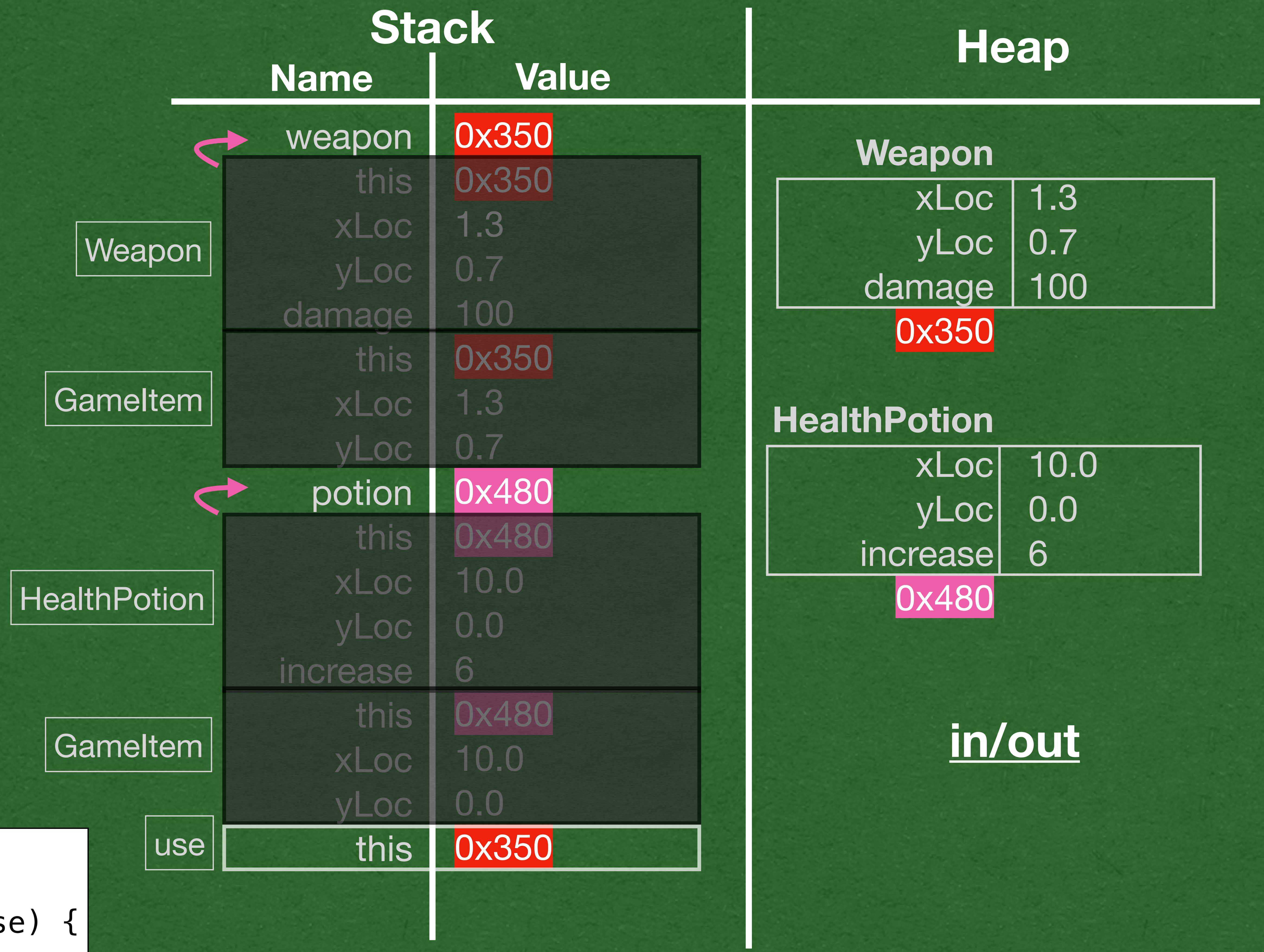
- Follow the type of the calling object!
- This call is from an object of type Weapon
  - Look in the Weapon class

```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    public void use() {
        System.out.println("Item Used");
    }
}
```

```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    @Override
    public void use() {
        System.out.println("Damage dealt: " + this.damage);
    }
}
```

```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
}
```

```java
public static void main(String[] args) {
    Weapon weapon = new Weapon(1.3, 0.7, 100);
    HealthPotion potion = new HealthPotion(10.0, 0.0, 6);
    weapon.use();
    potion.use();
}
```



Stack

| Name | Value |
|---|---|
| weapon | 0x350 |
| this | 0x350 |
| xLoc | 1.3 |
| yLoc | 0.7 |
| damage | 100 |
| this | 0x350 |
| xLoc | 1.3 |
| yLoc | 0.7 |
| potion | 0x480 |
| this | 0x480 |
| xLoc | 10.0 |
| yLoc | 0.0 |
| increase | 6 |
| this | 0x480 |
| xLoc | 10.0 |
| yLoc | 0.0 |
| this | 0x350 |

Heap

Weapon

| | |
|---|---|
| xLoc | 1.3 |
| yLoc | 0.7 |
| damage | 100 |

0x350

HealthPotion

| | |
|---|---|
| xLoc | 10.0 |
| yLoc | 0.0 |
| increase | 6 |

0x480

**in/out**

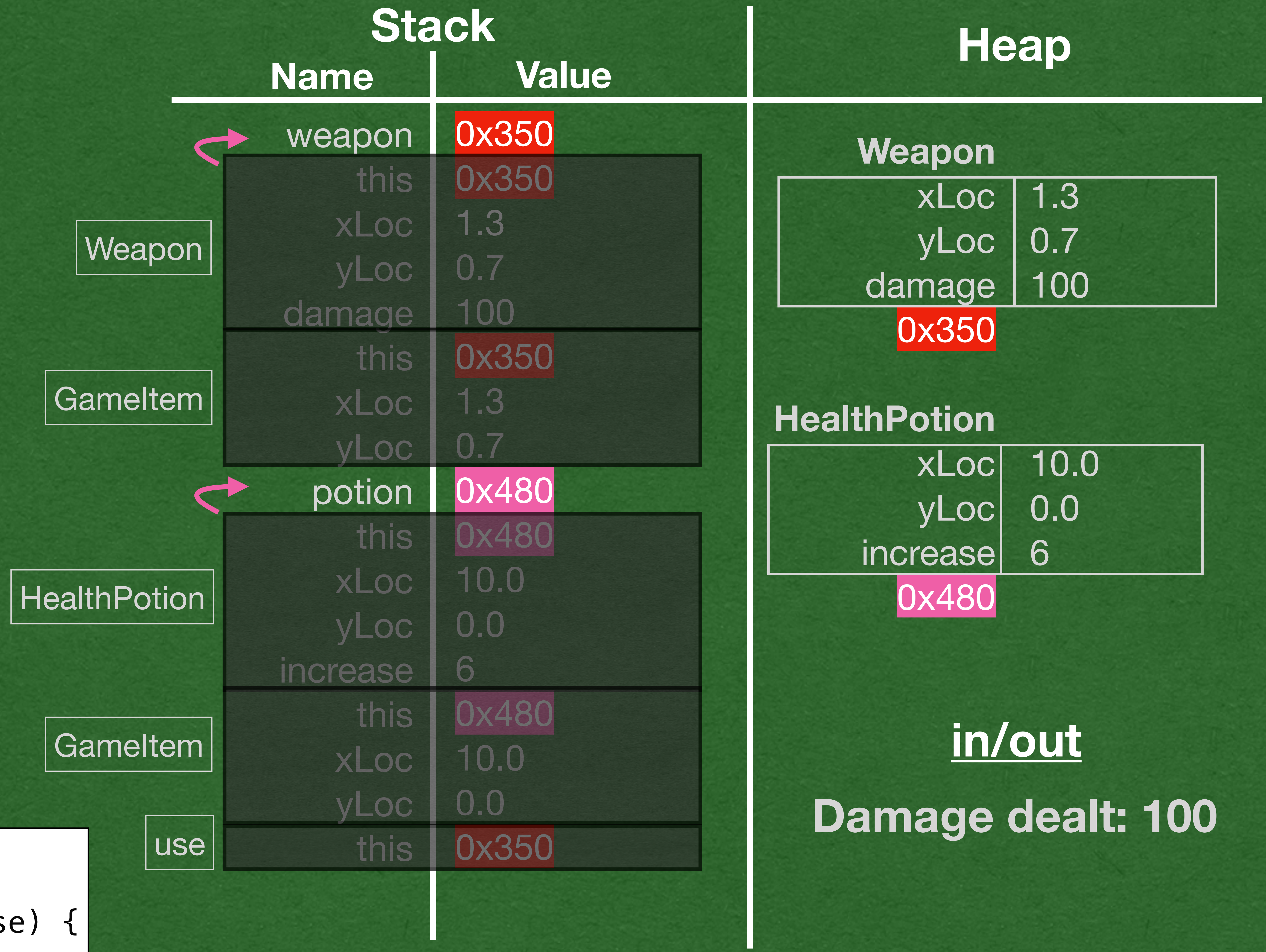
- We find a use method in the Weapon class
  - This is the method that's called

```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    public void use() {
        System.out.println("Item Used");
    }
}
```

```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    @Override
    public void use() {
        System.out.println("Damage dealt: " + this.damage);
    }
}
```

```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
}
```

```java
public static void main(String[] args) {
    Weapon weapon = new Weapon(1.3, 0.7, 100);
    HealthPotion potion = new HealthPotion(10.0, 0.0, 6);
    weapon.use();
    potion.use();
}
```



## Stack

| | Name | Value |
|---|---|---|
| | weapon | 0x350 |
| Weapon | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | damage | 100 |
| GameItem | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | potion | 0x480 |
| HealthPotion | this | 0x480 |
| | xLoc | 10.0 |
| | yLoc | 0.0 |
| | increase | 6 |
| GameItem | this | 0x480 |
| | xLoc | 10.0 |
| | yLoc | 0.0 |
| use | this | 0x350 |

## Heap

**Weapon**

| | |
|---|---|
| xLoc | 1.3 |
| yLoc | 0.7 |
| damage | 100 |

0x350

**HealthPotion**

| | |
|---|---|
| xLoc | 10.0 |
| yLoc | 0.0 |
| increase | 6 |

0x480

**in/out**

**Damage dealt: 100**

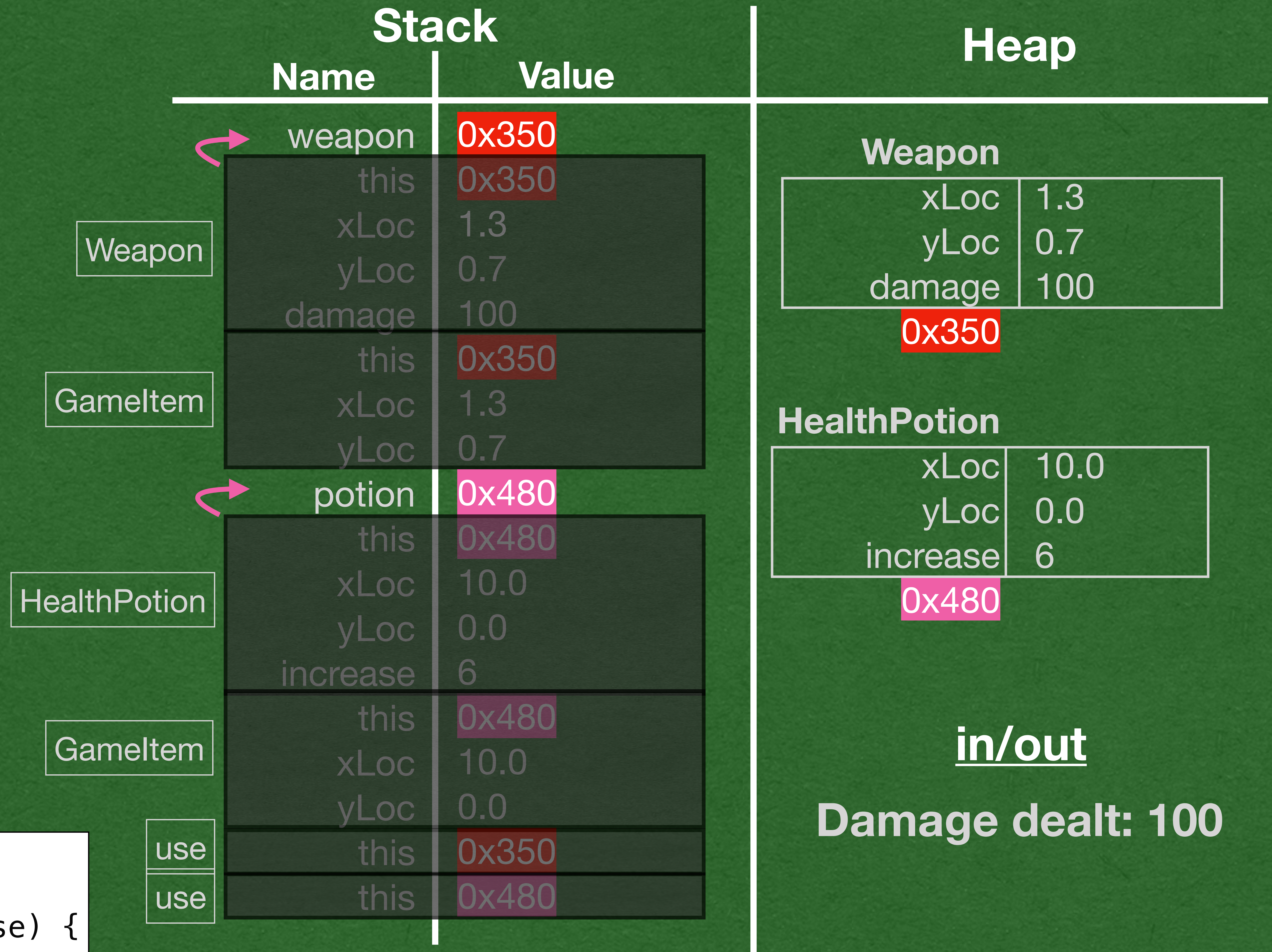
- Follow the same steps for the next call
- The calling object has type HealthPotion
  - Look in the HealthPotion class

```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    public void use() {
        System.out.println("Item Used");
    }
}
```

```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    @Override
    public void use() {
        System.out.println("Damage dealt: " + this.damage);
    }
}
```

```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
}
```

```java
public static void main(String[] args) {
    Weapon weapon = new Weapon(1.3, 0.7, 100);
    HealthPotion potion = new HealthPotion(10.0, 0.0, 6);
    weapon.use();
    potion.use();
}
```



## Stack

| | Name | Value |
|---|---|---|
| | weapon | 0x350 |
| Weapon | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | damage | 100 |
| GameItem | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | potion | 0x480 |
| HealthPotion | this | 0x480 |
| | xLoc | 10.0 |
| | yLoc | 0.0 |
| | increase | 6 |
| GameItem | this | 0x480 |
| | xLoc | 10.0 |
| | yLoc | 0.0 |
| use | this | 0x350 |
| use | this | 0x480 |

## Heap

**Weapon**

| | |
|---|---|
| xLoc | 1.3 |
| yLoc | 0.7 |
| damage | 100 |

0x350

**HealthPotion**

| | |
|---|---|
| xLoc | 10.0 |
| yLoc | 0.0 |
| increase | 6 |

0x480

**in/out**

Damage dealt: 100

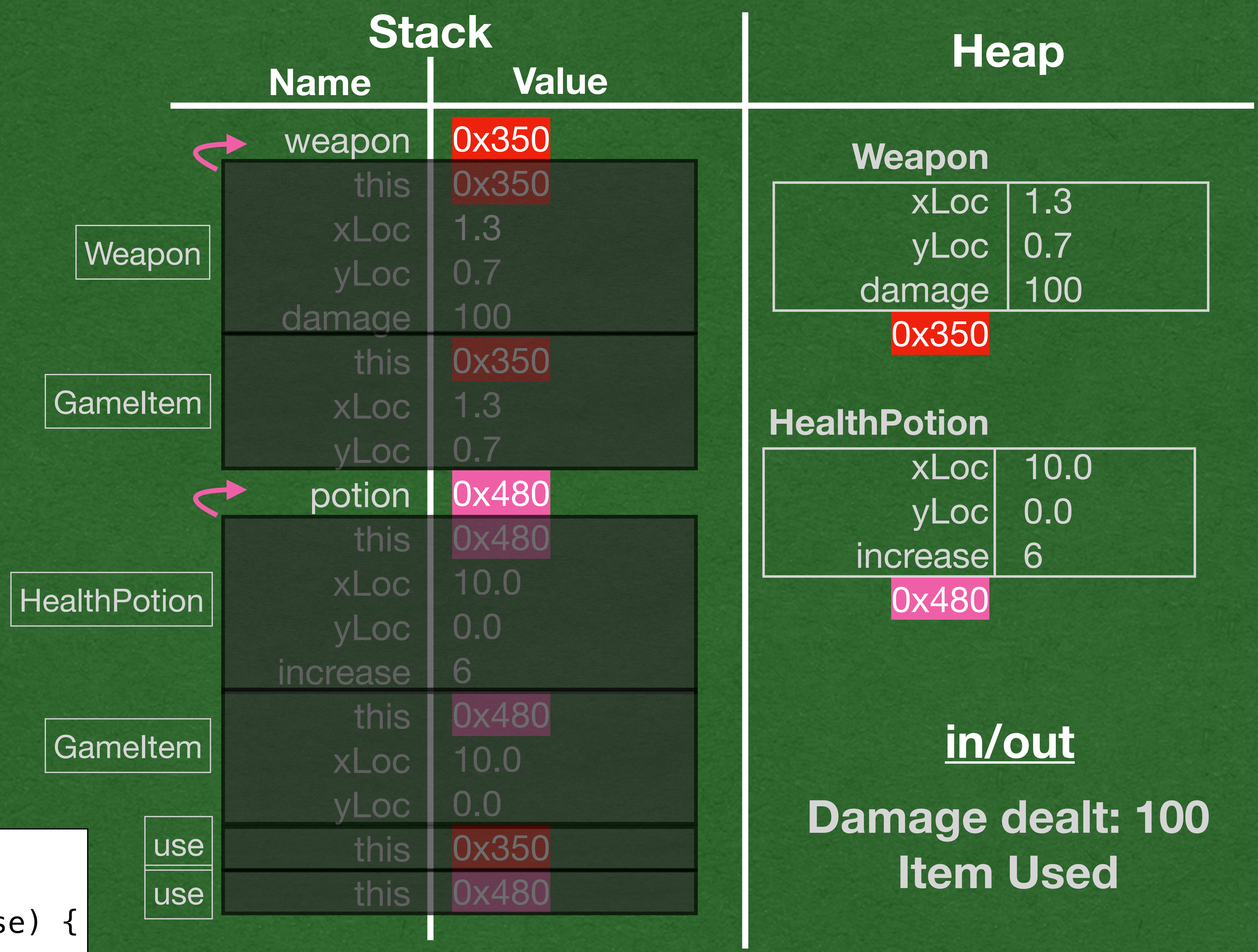
- We don't find a method named use in the HealthPotion class
- Continue our search in it's super class
  - We find and call the use method in the GameItem class

```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    public void use() {
        System.out.println("Item Used");
    }
}
```

```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    @Override
    public void use() {
        System.out.println("Damage dealt: " + this.damage);
    }
}
```

```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
}
```

```java
public static void main(String[] args) {
    Weapon weapon = new Weapon(1.3, 0.7, 100);
    HealthPotion potion = new HealthPotion(10.0, 0.0, 6);
    weapon.use();
    potion.use();
}
```



## Stack

| | Name | Value |
|---|---|---|
| | weapon | 0x350 |
| Weapon | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | damage | 100 |
| GameItem | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | potion | 0x480 |
| HealthPotion | this | 0x480 |
| | xLoc | 10.0 |
| | yLoc | 0.0 |
| | increase | 6 |
| GameItem | this | 0x480 |
| | xLoc | 10.0 |
| | yLoc | 0.0 |
| use | this | 0x350 |
| use | this | 0x480 |

## Heap

**Weapon**

| | |
|---|---|
| xLoc | 1.3 |
| yLoc | 0.7 |
| damage | 100 |

0x350

**HealthPotion**

| | |
|---|---|
| xLoc | 10.0 |
| yLoc | 0.0 |
| increase | 6 |

0x480

**in/out**

**Damage dealt: 100**
**Item Used**

- Weapon overrides the use method, so it's use method is called
- HealthPotion does not override the use method so the inherited use method is called

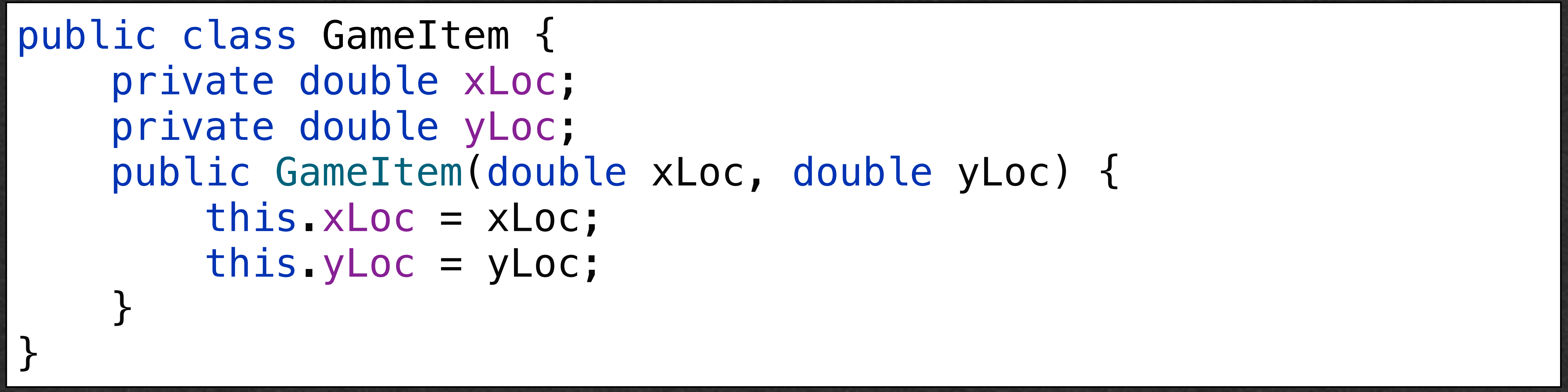
# toString

# The Object Class

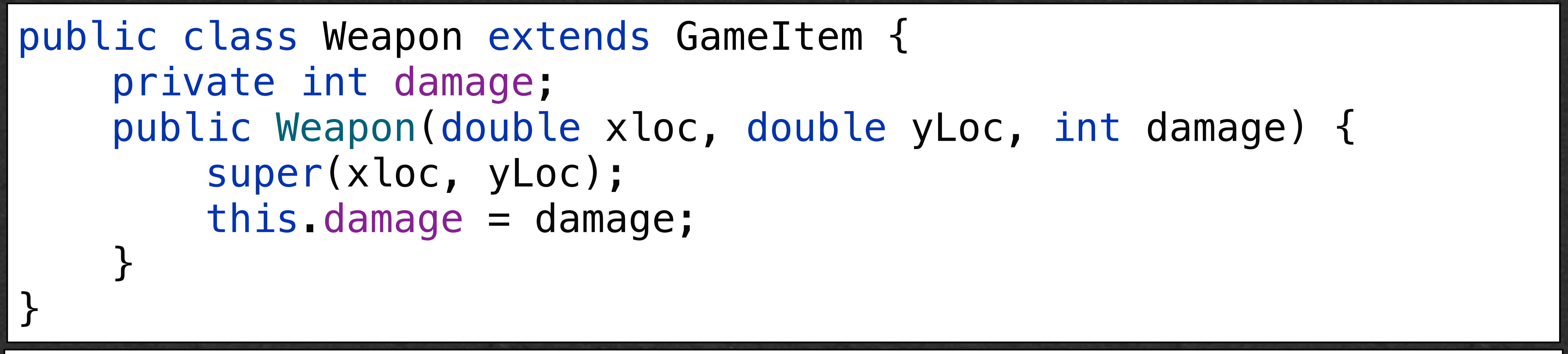
- Every class in Java extends Object either directly or indirectly
- Every object in Java has a toString and equals method that it inherited from Object
- We can override toString if we want custom behavior

# toString

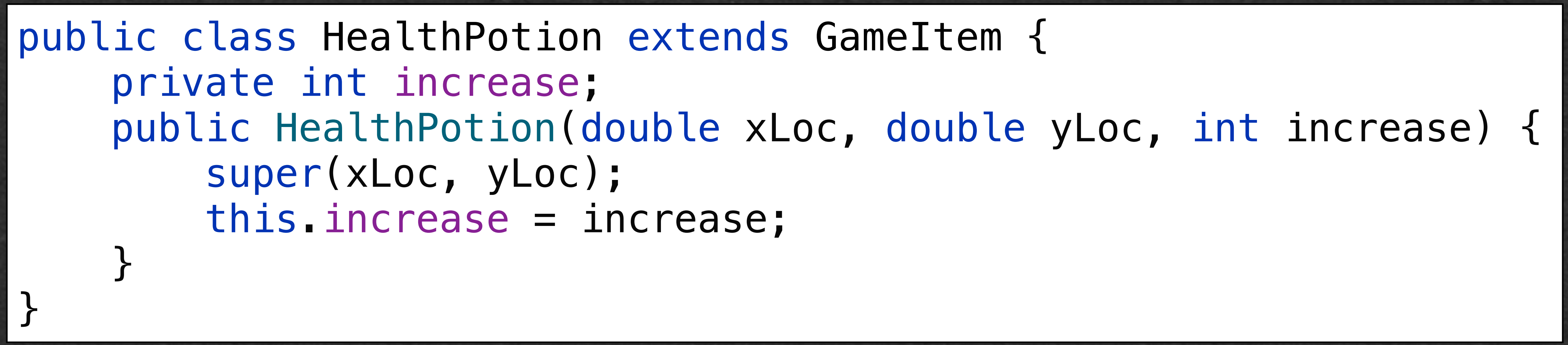
- When calling toString on HealthPotion or Weapon:
  - We don't find a toString method in the class matching the type of the object
  - Continue the search in GameItem
  - Don't find toString in GameItem
  - Continue the search in Object
  - Call the method defined in Object



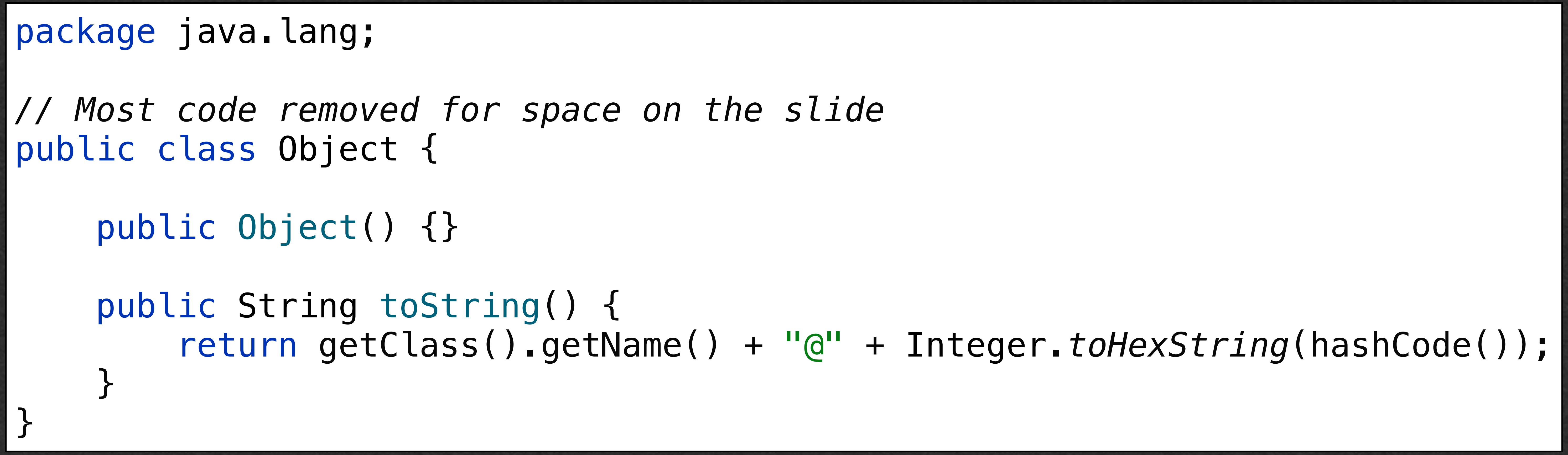
```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
}
```



```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
}
```



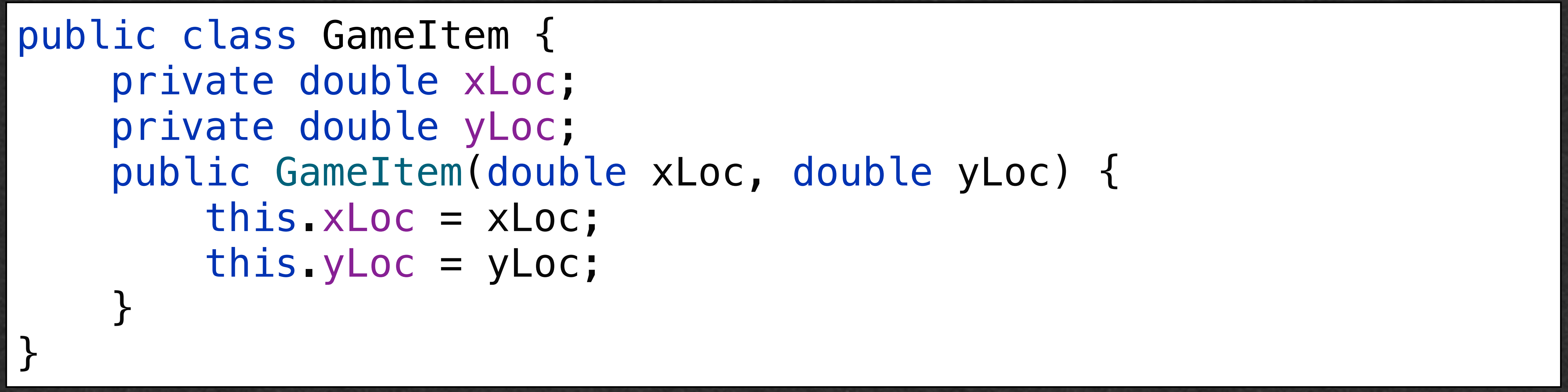
```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
}
```



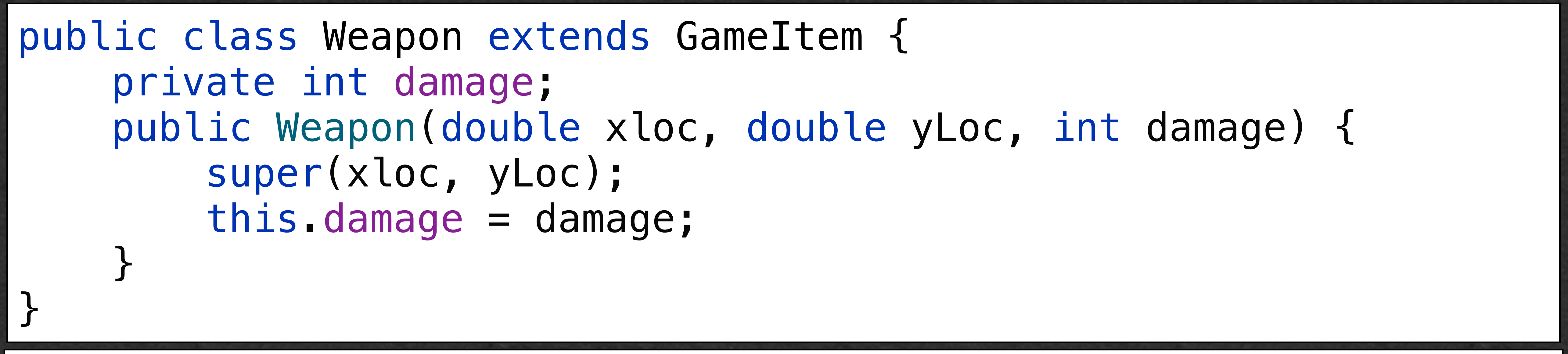
```java
package java.lang;

// Most code removed for space on the slide
public class Object {

    public Object() {}

    public String toString() {
        return getClass().getName() + "@" + Integer.toHexString(hashCode());
    }
}
```

# toString

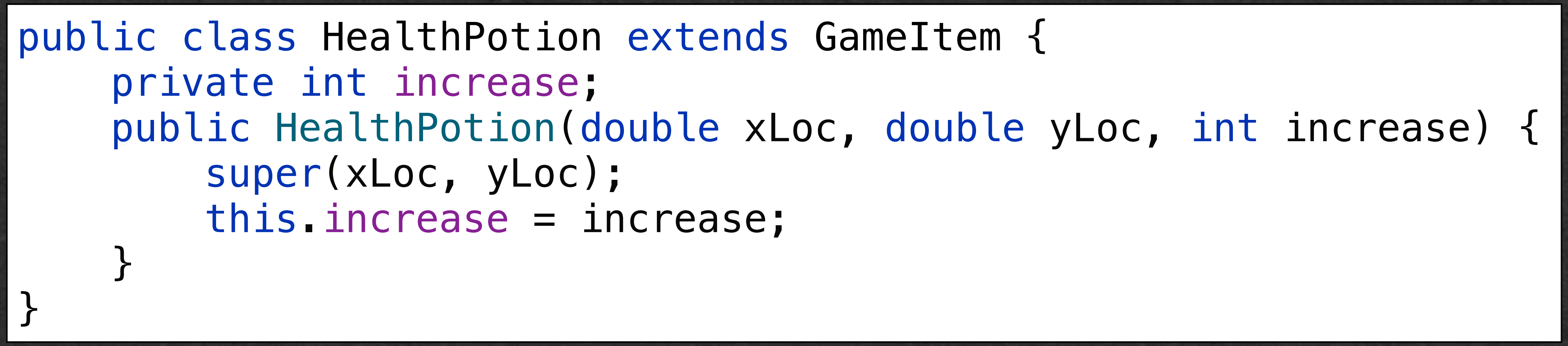
- The toString method inherited from the Object class will return:
  - {object_type}@{hex_value}
  - week6.Weapon@452b3a41
  - week6.HealthPotion@4a574795



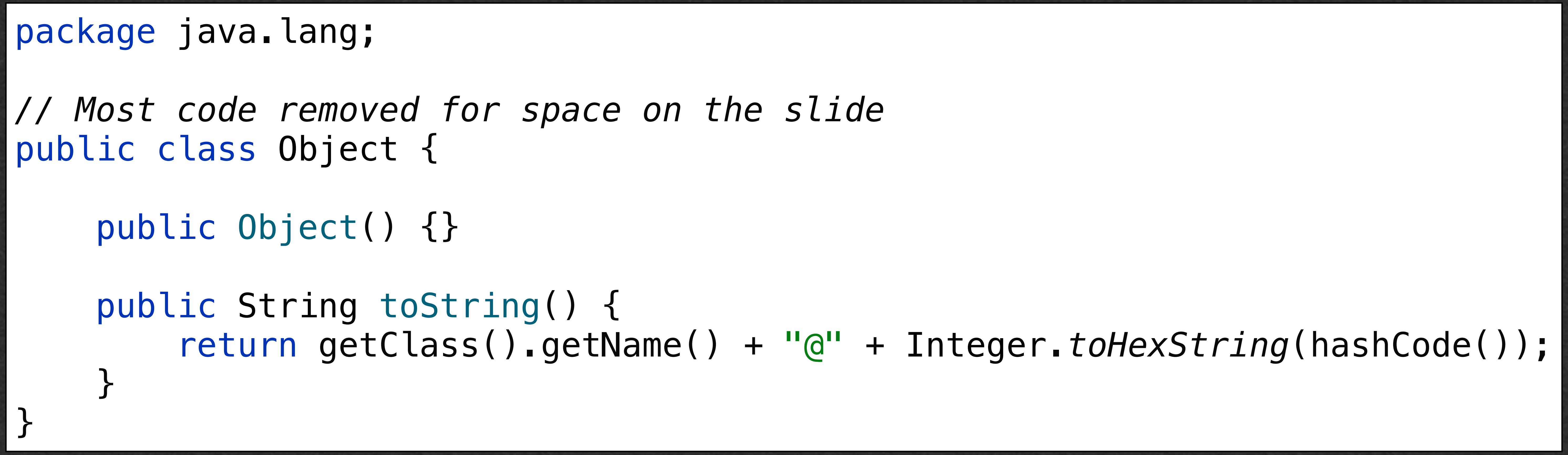
```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
}
```



```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
}
```



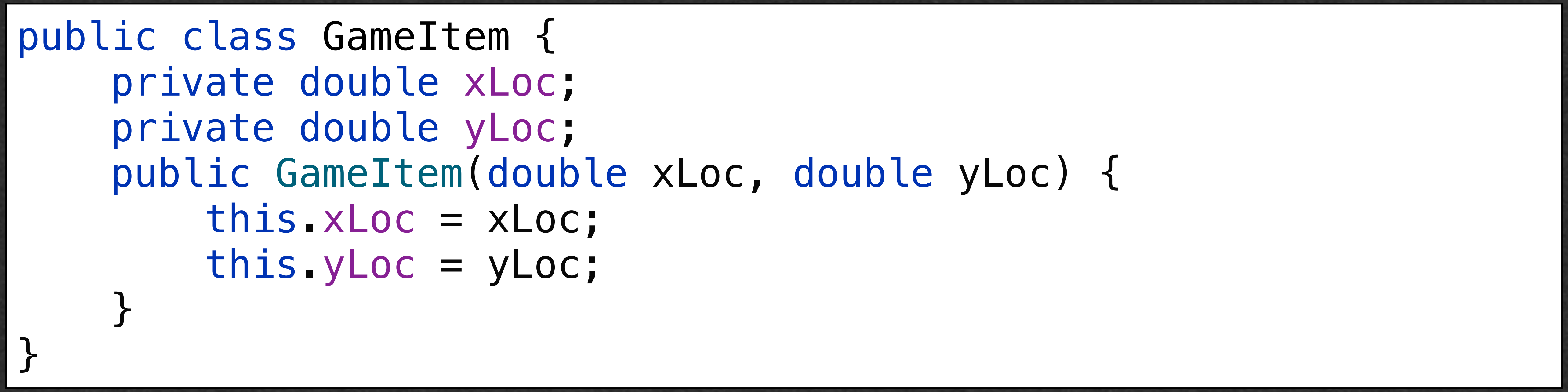
```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
}
```



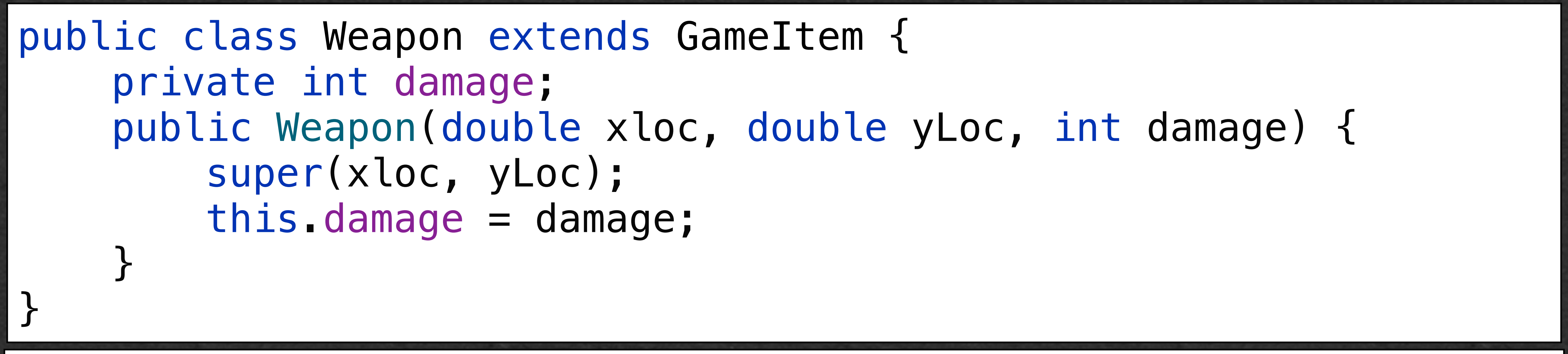
```java
package java.lang;

// Most code removed for space on the slide
public class Object {

    public Object() {}

    public String toString() {
        return getClass().getName() + "@" + Integer.toHexString(hashCode());
    }
}
```

# toString

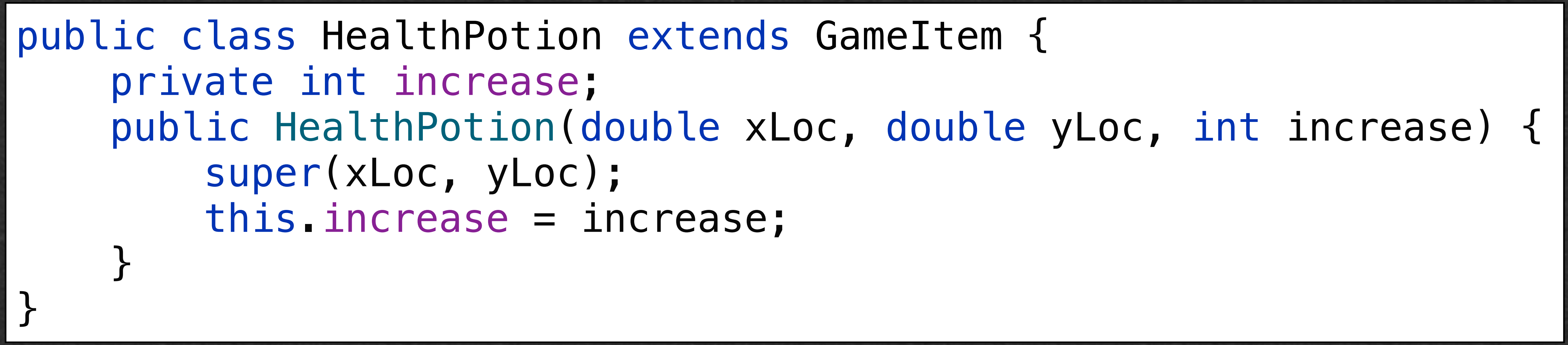
- The default behavior of toString is mostly useless
    - Even the official documentation says - "*It is recommended that all subclasses override this method.*"
- We will override this method



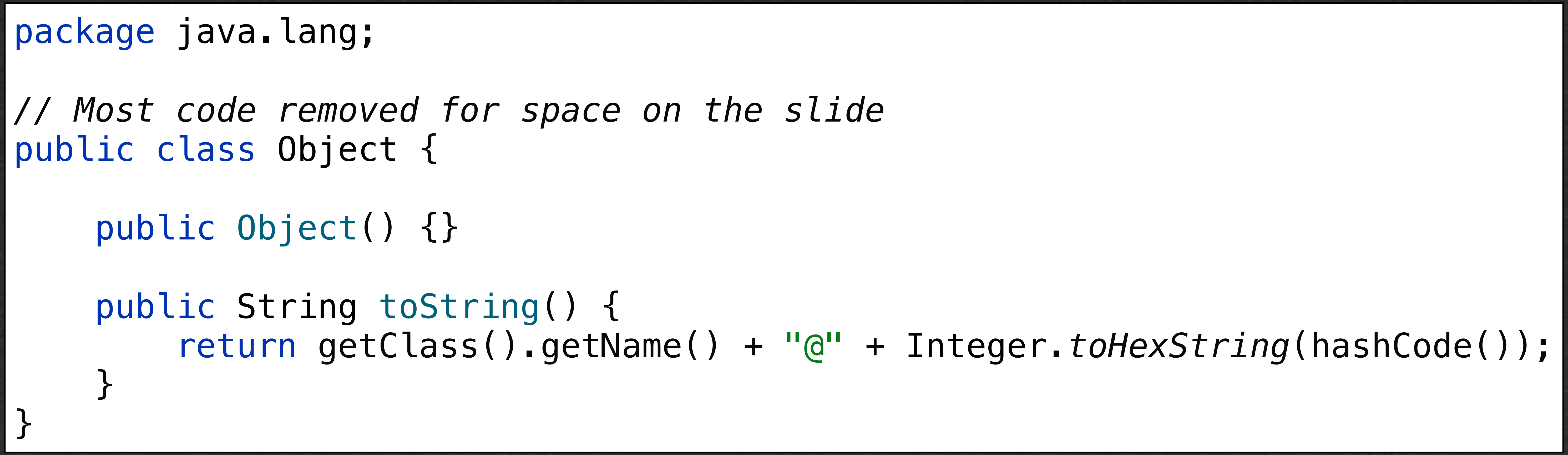
```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
}
```



```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
}
```



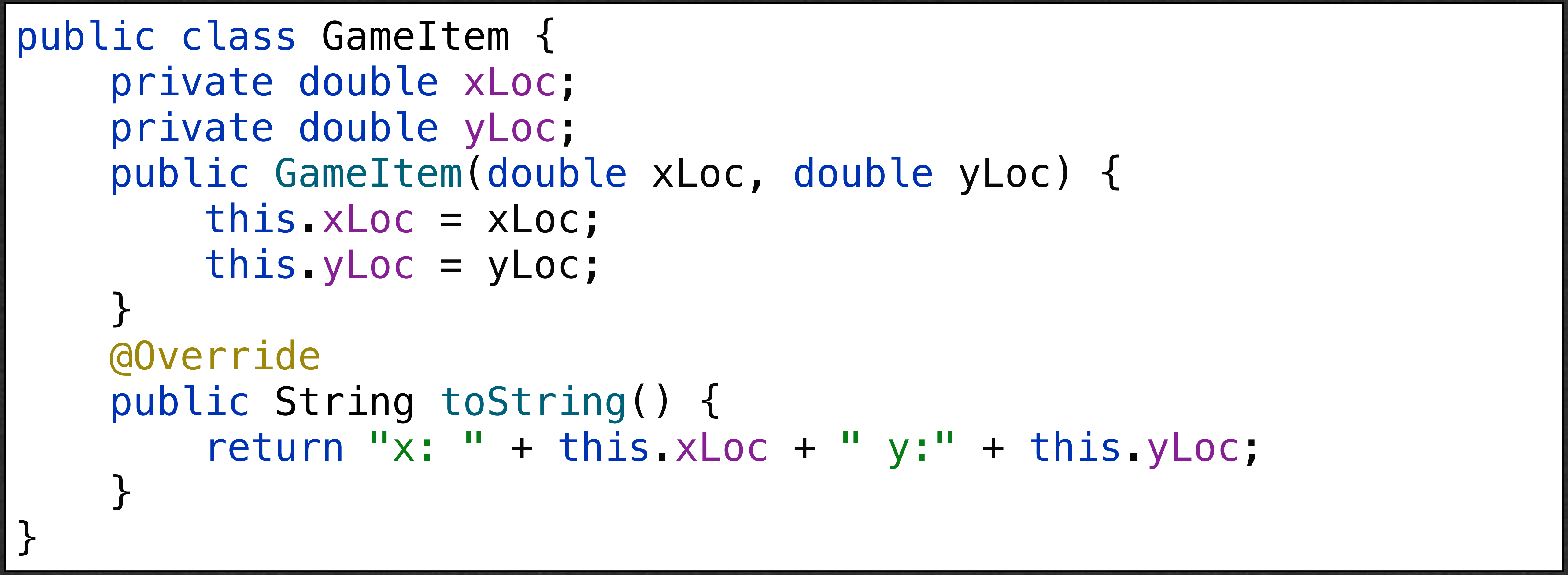
```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
}
```



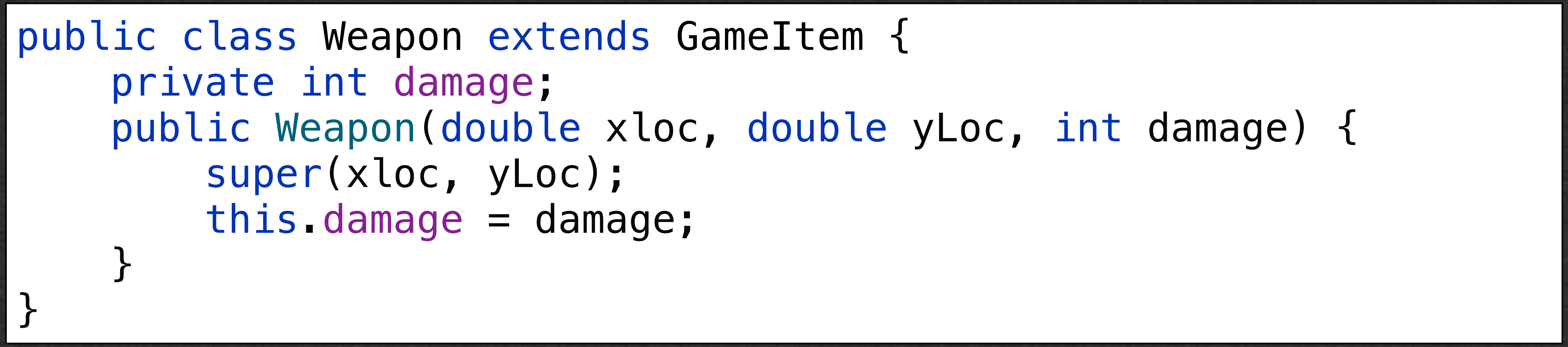
```java
package java.lang;

// Most code removed for space on the slide
public class Object {

    public Object() {}

    public String toString() {
        return getClass().getName() + "@" + Integer.toHexString(hashCode());
    }
}
```

# toString

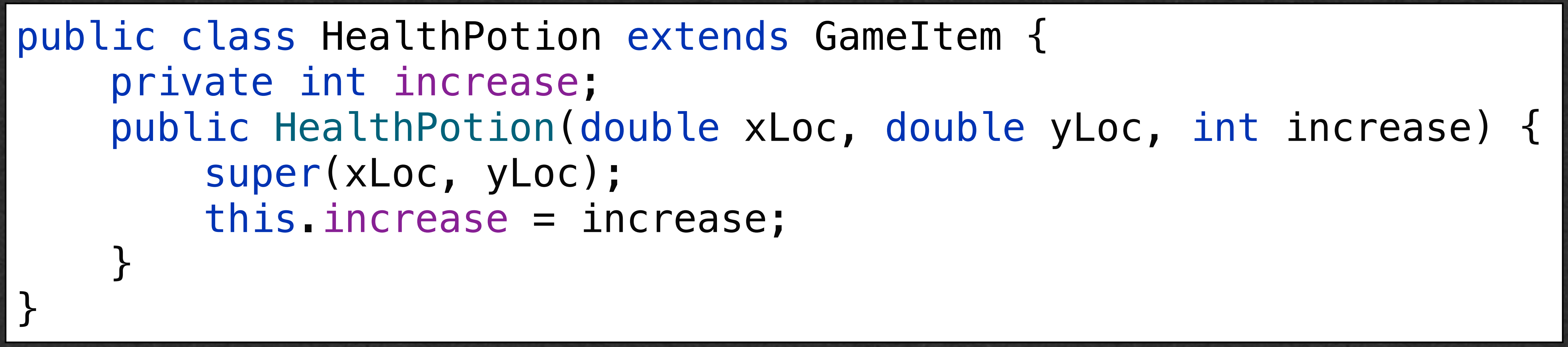
- GameItem implicitly extends Object and inherits toString
- We override this default behavior to return something meaningful to our GameItems
  - In previous lectures, we did this without the @Override annotation
- Weapon and HealthPotion inherit the override method from GameItem



```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    @Override
    public String toString() {
        return "x: " + this.xLoc + " y:" + this.yLoc;
    }
}
```



```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
}
```



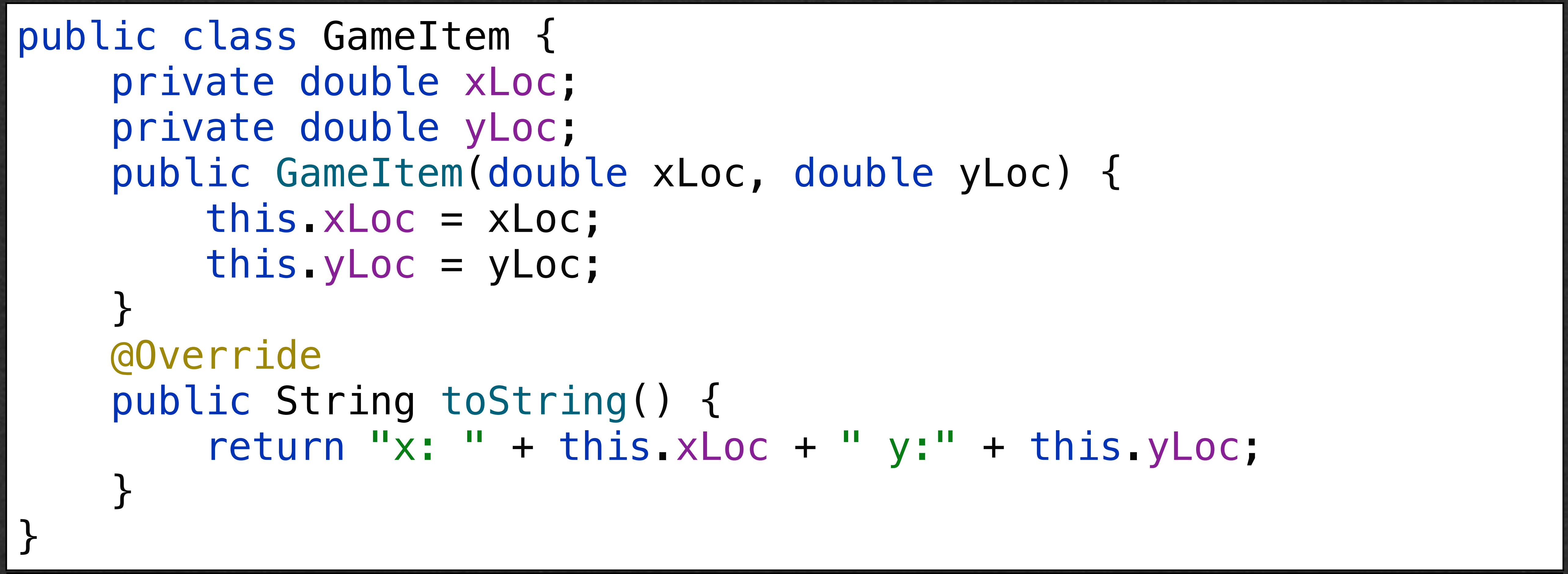
```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
}
```

# toString

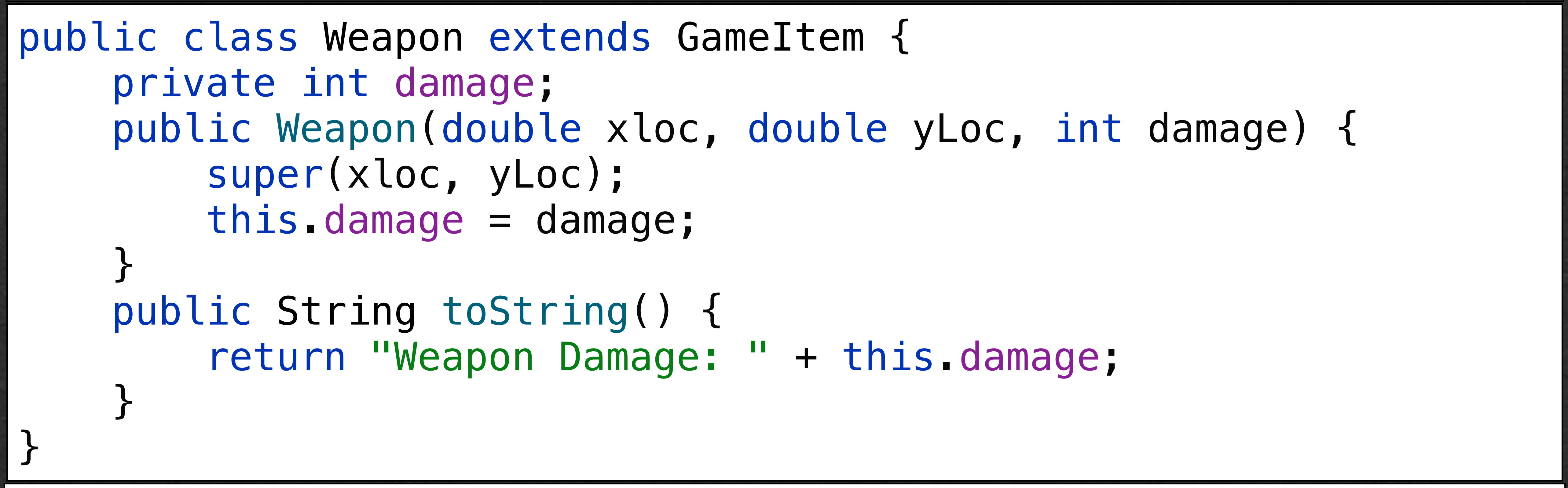
- We can also override a method that has already been overridden

- In both Weapon and HealthPotion
  - Override toString again to return Strings specific to each type

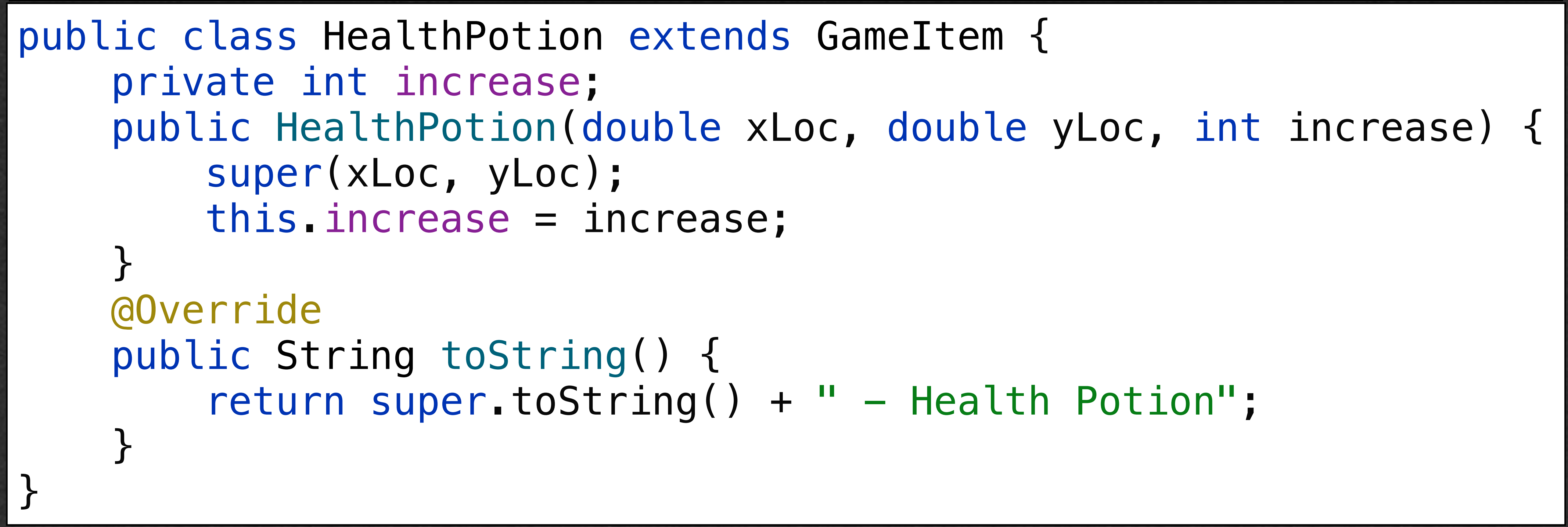
- Note: In Weapon we omitted the annotation and in HealthPotion we used the annotation
  - Both have the same result on our program
  - No reason to mix using and not using the annotation except for an example



```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    @Override
    public String toString() {
        return "x: " + this.xLoc + " y:" + this.yLoc;
    }
}
```



```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    public String toString() {
        return "Weapon Damage: " + this.damage;
    }
}
```



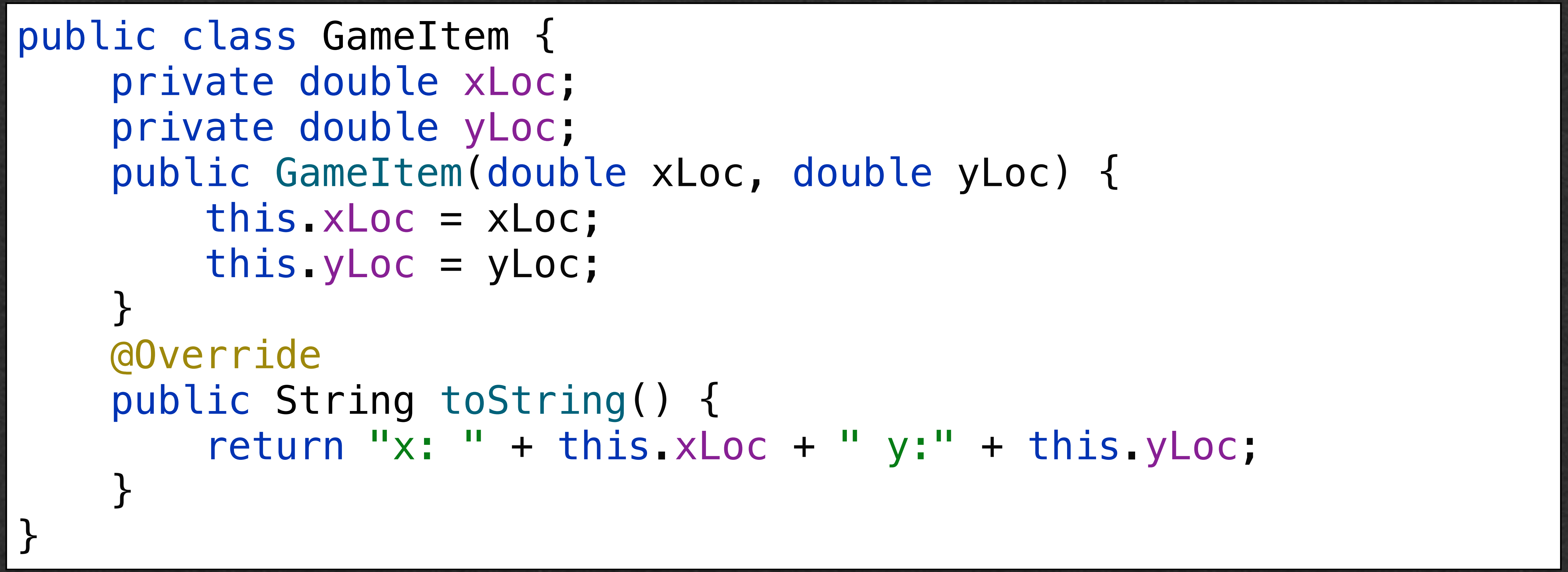
```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
    @Override
    public String toString() {
        return super.toString() + " - Health Potion";
    }
}
```

# super

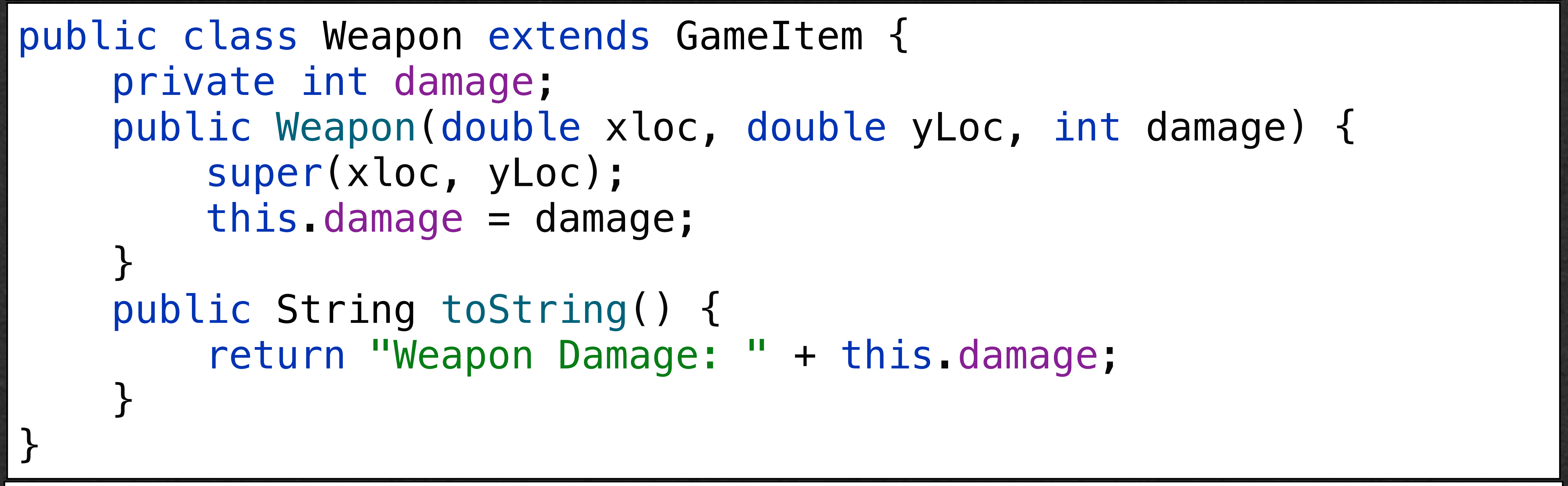
- We saw the super keyword when calling the super classes constructor

- Another use is to call an override method
  - Here, we call the GameItem's toString method

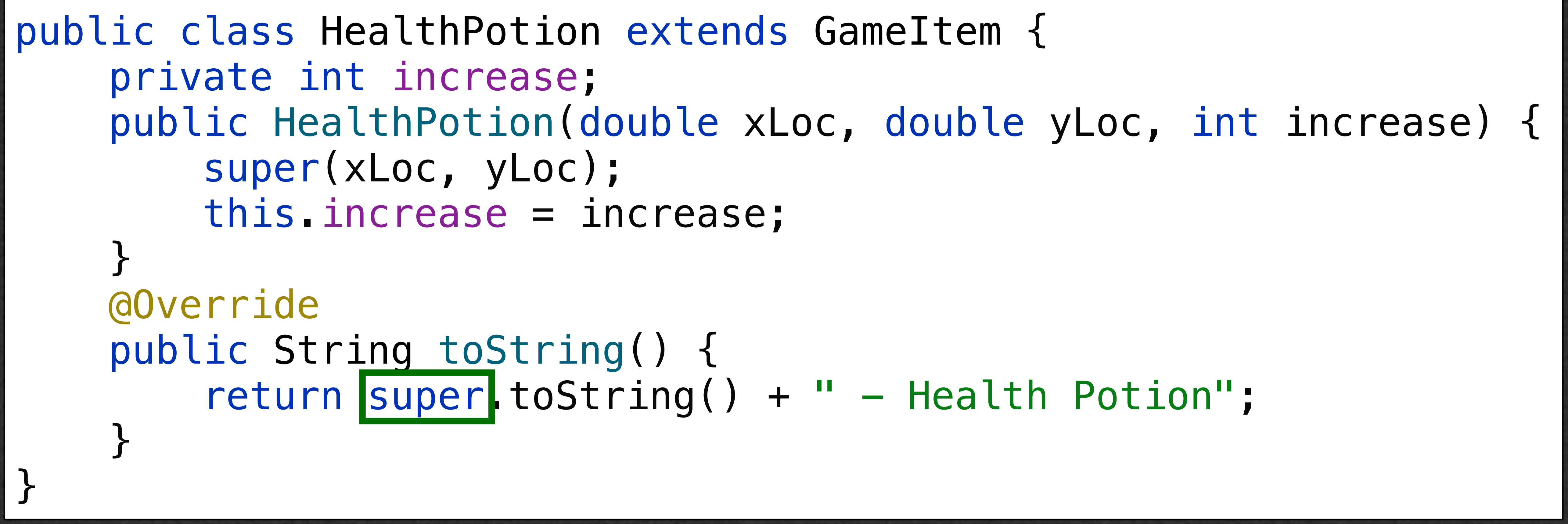
- It's common to add functionality to a method instead of completely replacing it
  - Override the method, but still call the method you are replacing with *super*



```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    @Override
    public String toString() {
        return "x: " + this.xLoc + " y:" + this.yLoc;
    }
}
```



```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    public String toString() {
        return "Weapon Damage: " + this.damage;
    }
}
```



```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
    @Override
    public String toString() {
        return super.toString() + " - Health Potion";
    }
}
```

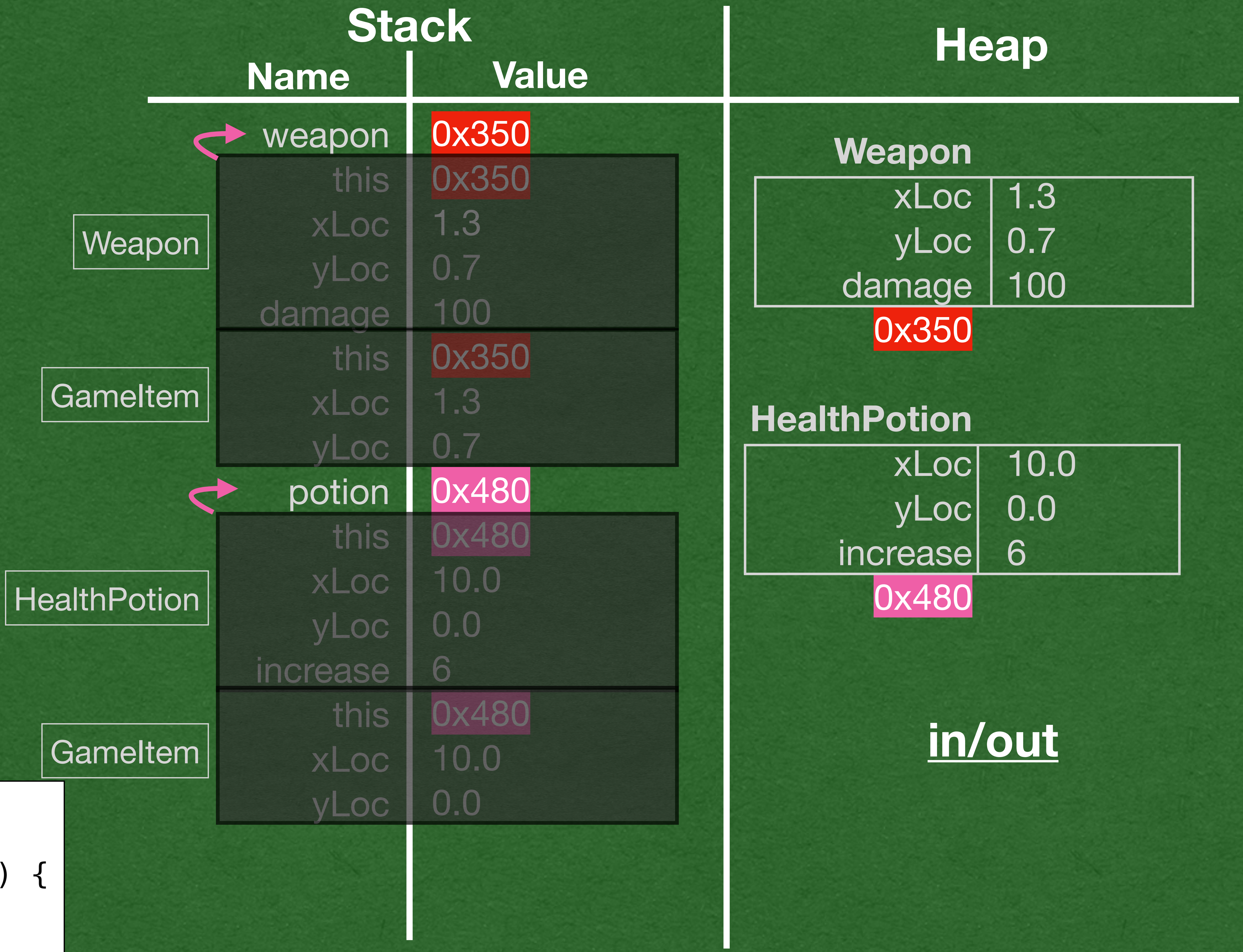
# Another Memory Diagram

```
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    @Override
    public String toString() {
        return "x: " + this.xLoc + " y:" + this.yLoc;
    }
}
```

```
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    public String toString() {
        return "Weapon Damage: " + this.damage;
    }
}
```

```
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
    @Override
    public String toString() {
        return super.toString() + " - Health Potion";
    }
}
```

```
    Weapon weapon = new Weapon(1.3, 0.7, 100);
    HealthPotion potion = new HealthPotion(10.0, 0.0, 6);
    System.out.println(weapon);
    System.out.println(potion);
```



## Stack

| | Name | Value |
|---|---|---|
| | weapon | 0x350 |
| Weapon | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | damage | 100 |
| GameItem | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | potion | 0x480 |
| HealthPotion | this | 0x480 |
| | xLoc | 10.0 |
| | yLoc | 0.0 |
| | increase | 6 |
| GameItem | this | 0x480 |
| | xLoc | 10.0 |
| | yLoc | 0.0 |

## Heap

Weapon

| | |
|---|---|
| xLoc | 1.3 |
| yLoc | 0.7 |
| damage | 100 |

0x350

HealthPotion

| | |
|---|---|
| xLoc | 10.0 |
| yLoc | 0.0 |
| increase | 6 |

0x480

**in/out**

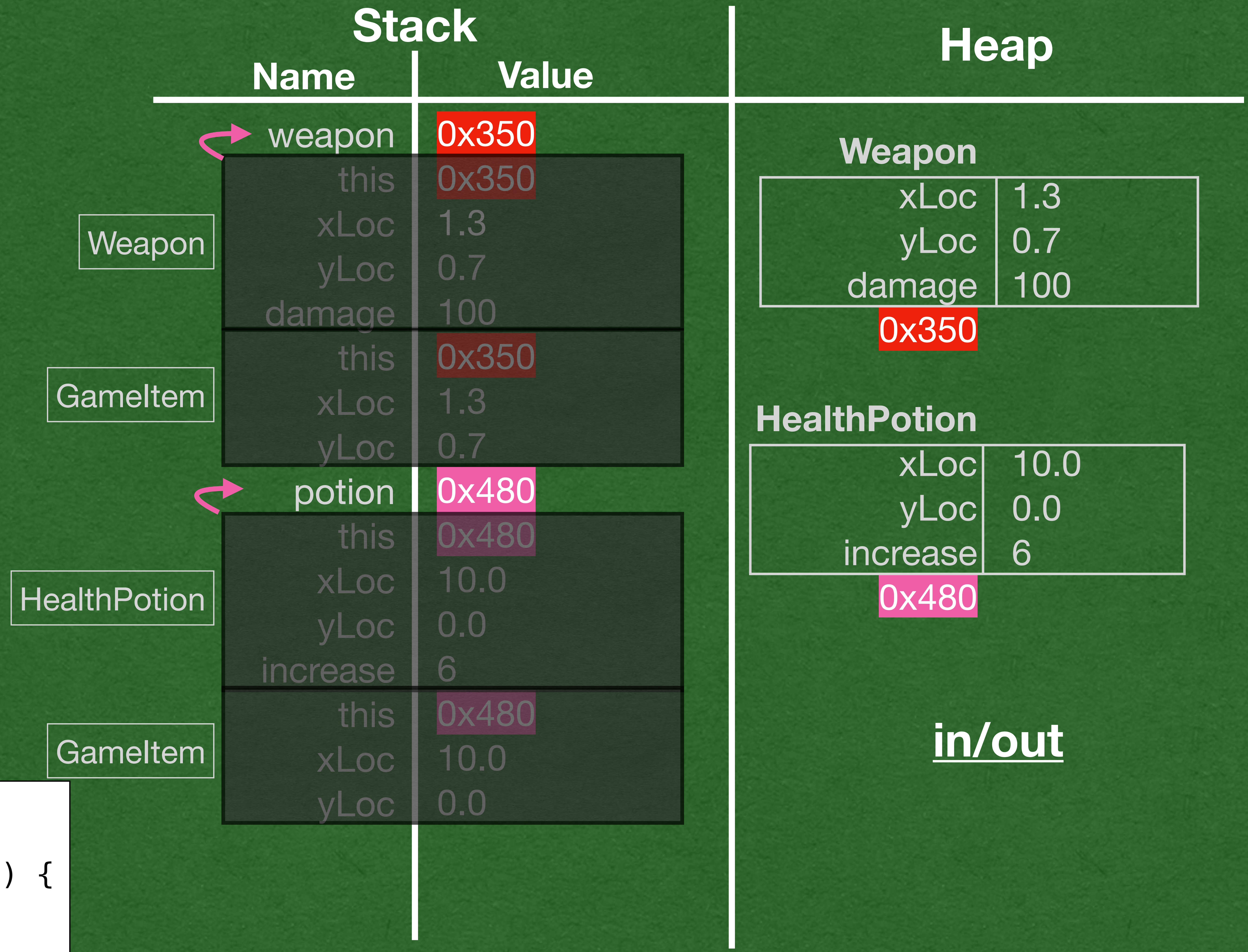
- What happens when we print Weapons and HealthPotions to the screen?

```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    @Override
    public String toString() {
        return "x: " + this.xLoc + " y:" + this.yLoc;
    }
}
```

```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    public String toString() {
        return "Weapon Damage: " + this.damage;
    }
}
```

```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
    @Override
    public String toString() {
        return super.toString() + " - Health Potion";
    }
}
```

```java
    Weapon weapon = new Weapon(1.3, 0.7, 100);
    HealthPotion potion = new HealthPotion(10.0, 0.0, 6);
    System.out.println(weapon);
    System.out.println(potion);
```



**Stack**

| | Name | Value |
|---|---|---|
| | weapon | 0x350 |
| Weapon | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | damage | 100 |
| GameItem | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | potion | 0x480 |
| HealthPotion | this | 0x480 |
| | xLoc | 10.0 |
| | yLoc | 0.0 |
| | increase | 6 |
| GameItem | this | 0x480 |
| | xLoc | 10.0 |
| | yLoc | 0.0 |

**Heap**

Weapon

| | |
|---|---|
| xLoc | 1.3 |
| yLoc | 0.7 |
| damage | 100 |

0x350

HealthPotion

| | |
|---|---|
| xLoc | 10.0 |
| yLoc | 0.0 |
| increase | 6 |

0x480

**in/out**

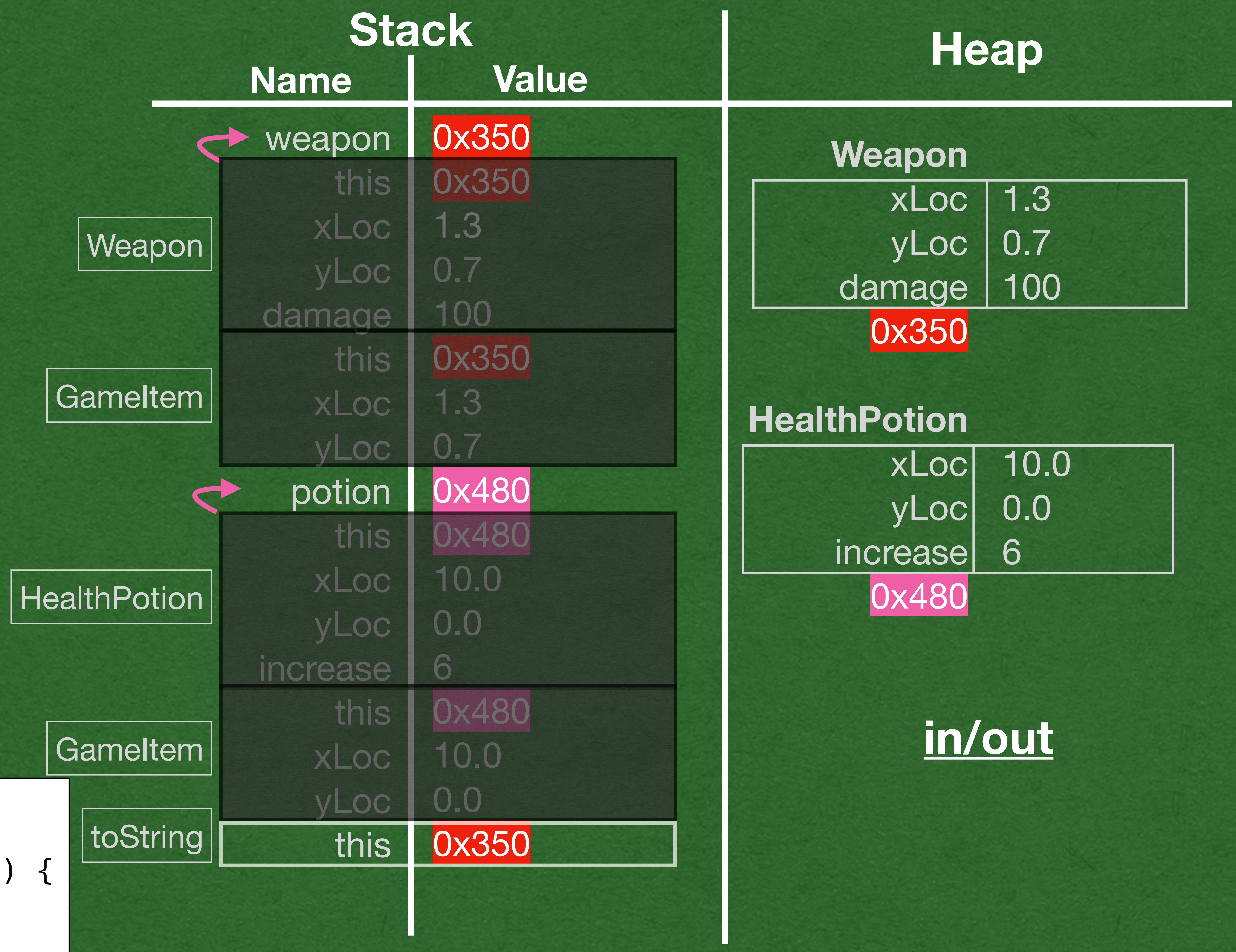
- System.out.println will call toString
  - You must call toString in your memory diagrams if you have the code for a toString method

```
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    @Override
    public String toString() {
        return "x: " + this.xLoc + " y:" + this.yLoc;
    }
}
```

```
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
  ➔public String toString() {
        return "Weapon Damage: " + this.damage;
    }
}
```

```
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
    @Override
    public String toString() {
        return super.toString() + " - Health Potion";
    }
}
```

```
    Weapon weapon = new Weapon(1.3, 0.7, 100);
    HealthPotion potion = new HealthPotion(10.0, 0.0, 6);
  ➔System.out.println(weapon);
    System.out.println(potion);
```



## Stack

|  | Name | Value |
|---|---|---|
|  | weapon | 0x350 |
| Weapon | this | 0x350 |
|  | xLoc | 1.3 |
|  | yLoc | 0.7 |
|  | damage | 100 |
| GameItem | this | 0x350 |
|  | xLoc | 1.3 |
|  | yLoc | 0.7 |
|  | potion | 0x480 |
| HealthPotion | this | 0x480 |
|  | xLoc | 10.0 |
|  | yLoc | 0.0 |
|  | increase | 6 |
| GameItem | this | 0x480 |
|  | xLoc | 10.0 |
|  | yLoc | 0.0 |
| toString | this | 0x350 |

## Heap

Weapon

|  |  |
|---|---|
| xLoc | 1.3 |
| yLoc | 0.7 |
| damage | 100 |

0x350

HealthPotion

|  |  |
|---|---|
| xLoc | 10.0 |
| yLoc | 0.0 |
| increase | 6 |

0x480

**in/out**

- The calling object has type Weapon
- Find toString in the Weapon class
  - The Override is implicit since there's no annotation

```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    @Override
    public String toString() {
        return "x: " + this.xLoc + " y:" + this.yLoc;
    }
}
```

```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    public String toString() {
        return "Weapon Damage: " + this.damage;
    }
}
```

```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
    @Override
    public String toString() {
        return super.toString() + " - Health Potion";
    }
}
```

```java
    Weapon weapon = new Weapon(1.3, 0.7, 100);
    HealthPotion potion = new HealthPotion(10.0, 0.0, 6);
    System.out.println(weapon);
    System.out.println(potion);
```

## Stack

|  | Name | Value |
|---|---|---|
|  | weapon | 0x350 |
| Weapon | this | 0x350 |
|  | xLoc | 1.3 |
|  | yLoc | 0.7 |
|  | damage | 100 |
| GameItem | this | 0x350 |
|  | xLoc | 1.3 |
|  | yLoc | 0.7 |
|  | potion | 0x480 |
| HealthPotion | this | 0x480 |
|  | xLoc | 10.0 |
|  | yLoc | 0.0 |
|  | increase | 6 |
| GameItem | this | 0x480 |
|  | xLoc | 10.0 |
|  | yLoc | 0.0 |
| toString | this | 0x350 |
| toString | this | 0x480 |

## Heap

**Weapon**

|  |  |
|---|---|
| xLoc | 1.3 |
| yLoc | 0.7 |
| damage | 100 |

0x350

**HealthPotion**

|  |  |
|---|---|
| xLoc | 10.0 |
| yLoc | 0.0 |
| increase | 6 |

0x480

## in/out

```
Weapon Damage: 100
```

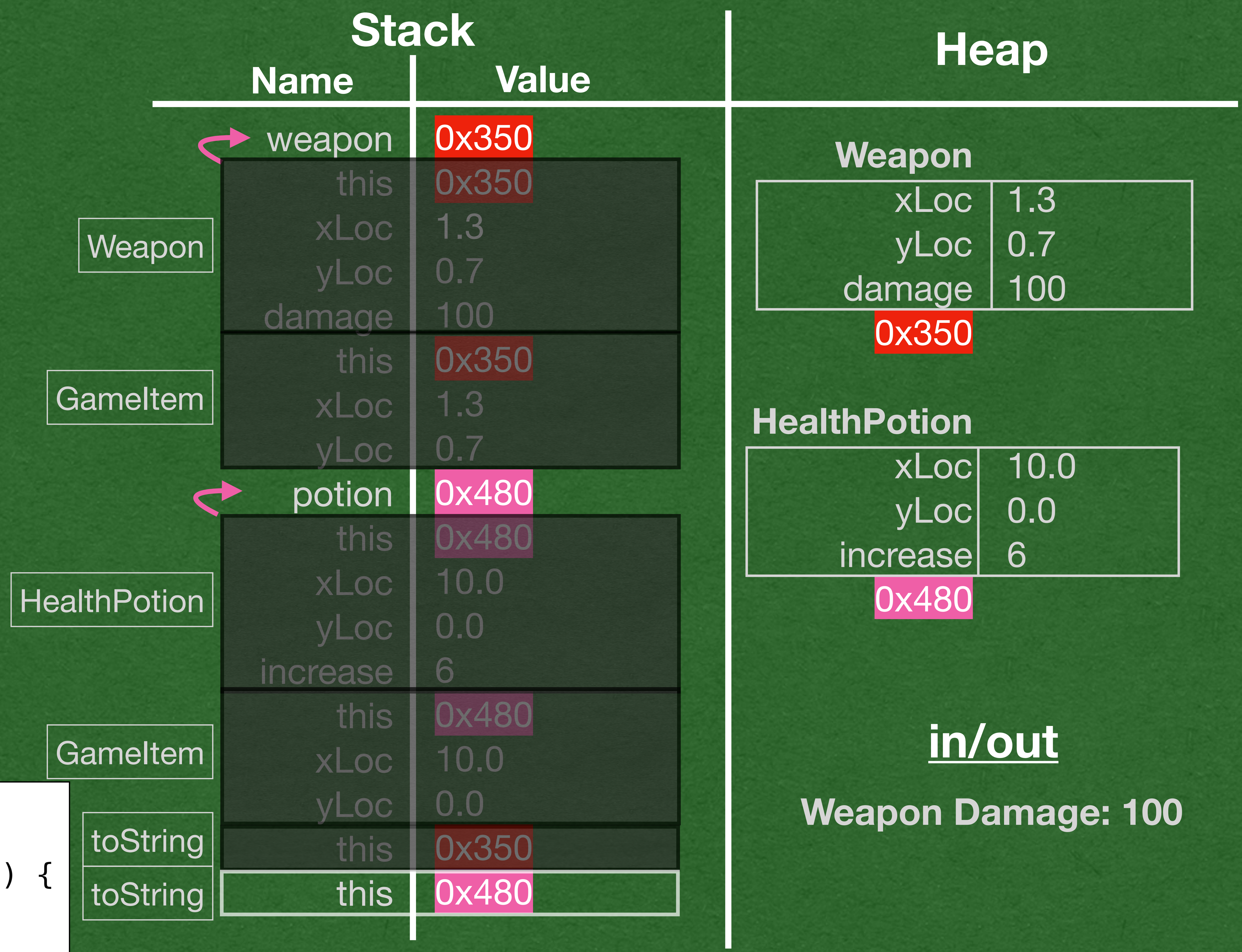
- Similar for HealthPotion
- Look in the HealthPotion class and find a toString method
  - This time the Override is explicit with an annotation

```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    @Override
    public String toString() {
        return "x: " + this.xLoc + " y:" + this.yLoc;
    }
}
```

```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    public String toString() {
        return "Weapon Damage: " + this.damage;
    }
}
```

```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
    @Override
    public String toString() {
        return super.toString() + " - Health Potion";
    }
}
```

```java
    Weapon weapon = new Weapon(1.3, 0.7, 100);
    HealthPotion potion = new HealthPotion(10.0, 0.0, 6);
    System.out.println(weapon);
    System.out.println(potion);
```



## Stack

| | Name | Value |
|---|---|---|
| | weapon | 0x350 |
| Weapon | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | damage | 100 |
| GameItem | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | potion | 0x480 |
| HealthPotion | this | 0x480 |
| | xLoc | 10.0 |
| | yLoc | 0.0 |
| | increase | 6 |
| GameItem | this | 0x480 |
| | xLoc | 10.0 |
| | yLoc | 0.0 |
| toString | this | 0x350 |
| toString | this | 0x480 |

## Heap

Weapon

| | |
|---|---|
| xLoc | 1.3 |
| yLoc | 0.7 |
| damage | 100 |

0x350

HealthPotion

| | |
|---|---|
| xLoc | 10.0 |
| yLoc | 0.0 |
| increase | 6 |

0x480

**in/out**

Weapon Damage: 100

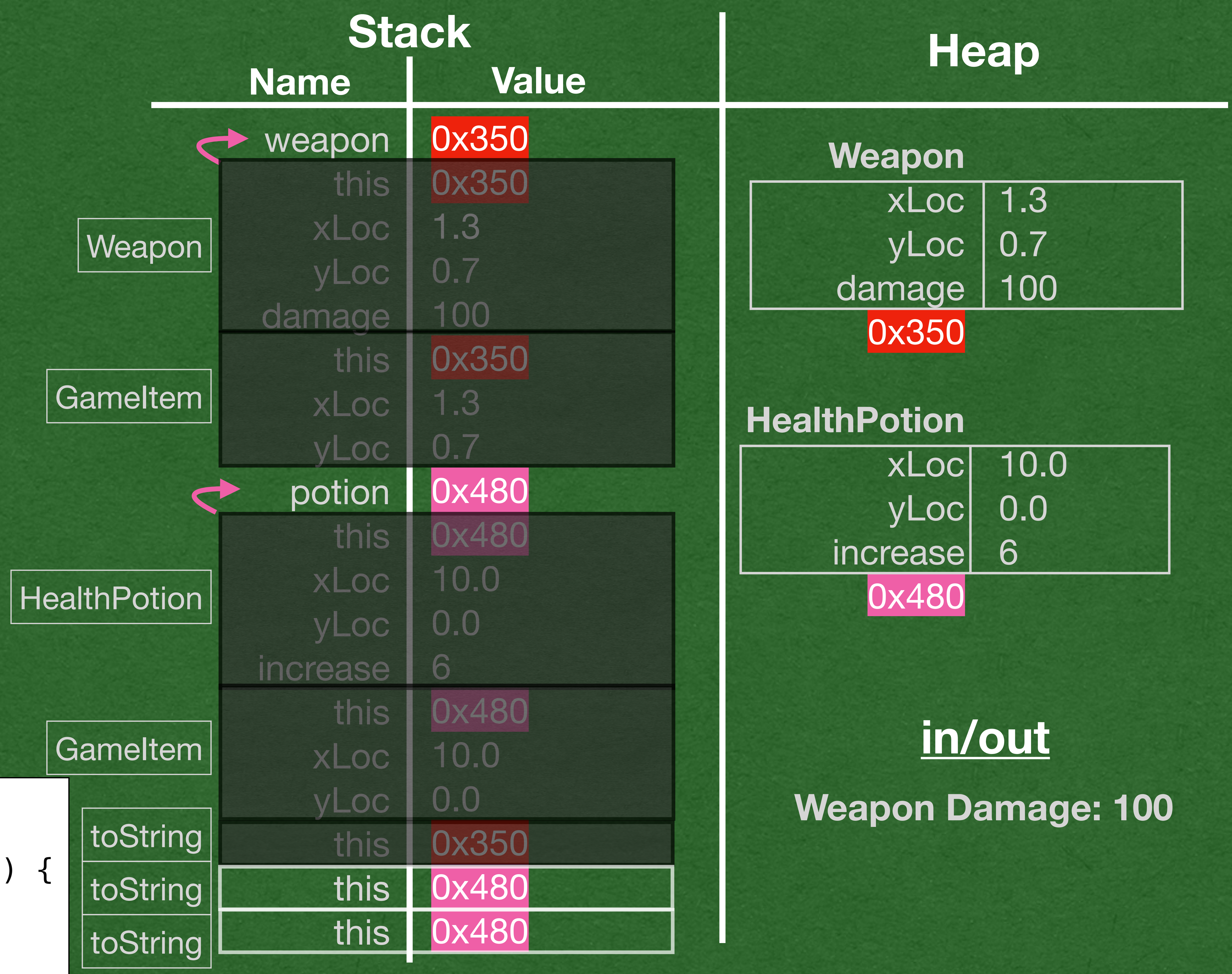
- For a super method call:
  - Look in the super class for a matching method

```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    @Override
    public String toString() {
        return "x: " + this.xLoc + " y:" + this.yLoc;
    }
}
```

```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    public String toString() {
        return "Weapon Damage: " + this.damage;
    }
}
```

```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
    @Override
    public String toString() {
        return super.toString() + " - Health Potion";
    }
}
```

```java
    Weapon weapon = new Weapon(1.3, 0.7, 100);
    HealthPotion potion = new HealthPotion(10.0, 0.0, 6);
    System.out.println(weapon);
    System.out.println(potion);
```



## Stack

| | Name | Value |
|---|---|---|
| | weapon | 0x350 |
| Weapon | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | damage | 100 |
| GameItem | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | potion | 0x480 |
| HealthPotion | this | 0x480 |
| | xLoc | 10.0 |
| | yLoc | 0.0 |
| | increase | 6 |
| GameItem | this | 0x480 |
| | xLoc | 10.0 |
| | yLoc | 0.0 |
| toString | this | 0x350 |
| toString | this | 0x480 |
| toString | this | 0x480 |

## Heap

**Weapon**

| | |
|---|---|
| xLoc | 1.3 |
| yLoc | 0.7 |
| damage | 100 |

0x350

**HealthPotion**

| | |
|---|---|
| xLoc | 10.0 |
| yLoc | 0.0 |
| increase | 6 |

0x480

**in/out**

**Weapon Damage: 100**

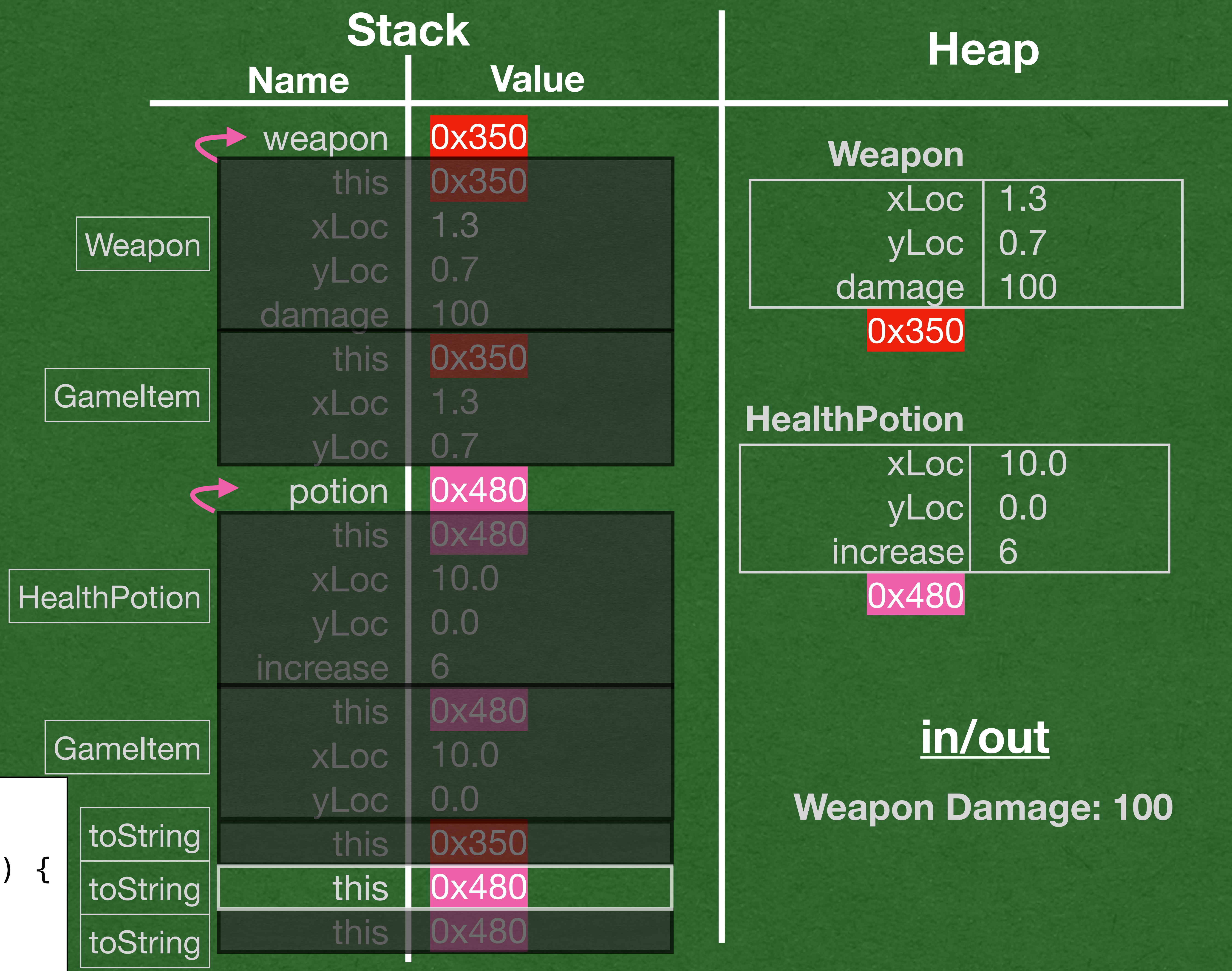
- We find a toString method in GameItem
  - This is the method called from super.toString
  - this in a super method call is the same as the original calling object

```java
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    @Override
    public String toString() {
        return "x: " + this.xLoc + " y:" + this.yLoc;
    }
}
```

```java
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    public String toString() {
        return "Weapon Damage: " + this.damage;
    }
}
```

```java
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
    @Override
    public String toString() {
        return super.toString() + " - Health Potion";
    }
}
```

```java
    Weapon weapon = new Weapon(1.3, 0.7, 100);
    HealthPotion potion = new HealthPotion(10.0, 0.0, 6);
    System.out.println(weapon);
    System.out.println(potion);
```



## Stack

|  | Name | Value |
|---|---|---|
|  | weapon | 0x350 |
| Weapon | this | 0x350 |
|  | xLoc | 1.3 |
|  | yLoc | 0.7 |
|  | damage | 100 |
| GameItem | this | 0x350 |
|  | xLoc | 1.3 |
|  | yLoc | 0.7 |
|  | potion | 0x480 |
| HealthPotion | this | 0x480 |
|  | xLoc | 10.0 |
|  | yLoc | 0.0 |
|  | increase | 6 |
| GameItem | this | 0x480 |
|  | xLoc | 10.0 |
|  | yLoc | 0.0 |
| toString | this | 0x350 |
| toString | this | 0x480 |
| toString | this | 0x480 |

## Heap

Weapon

| | |
|---|---|
| xLoc | 1.3 |
| yLoc | 0.7 |
| damage | 100 |

0x350

HealthPotion

| | |
|---|---|
| xLoc | 10.0 |
| yLoc | 0.0 |
| increase | 6 |

0x480

## in/out

Weapon Damage: 100

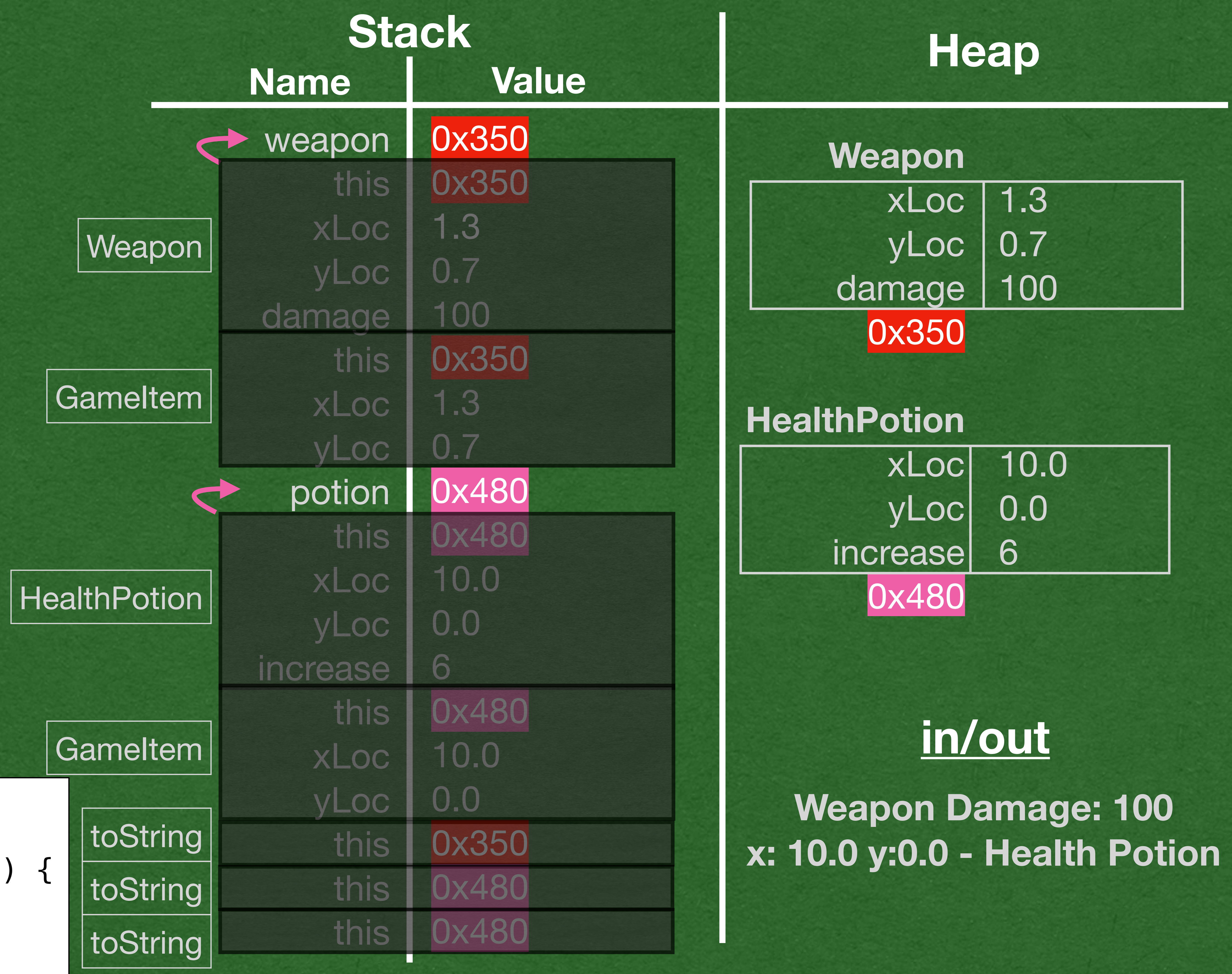
- The super method call returns "x: 10.0 y:0.0"
- The HealthPotion methods concatenates to this and returns

```
public class GameItem {
    private double xLoc;
    private double yLoc;
    public GameItem(double xLoc, double yLoc) {
        this.xLoc = xLoc;
        this.yLoc = yLoc;
    }
    @Override
    public String toString() {
        return "x: " + this.xLoc + " y:" + this.yLoc;
    }
}
```

```
public class Weapon extends GameItem {
    private int damage;
    public Weapon(double xloc, double yLoc, int damage) {
        super(xloc, yLoc);
        this.damage = damage;
    }
    public String toString() {
        return "Weapon Damage: " + this.damage;
    }
}
```

```
public class HealthPotion extends GameItem {
    private int increase;
    public HealthPotion(double xLoc, double yLoc, int increase) {
        super(xLoc, yLoc);
        this.increase = increase;
    }
    @Override
    public String toString() {
        return super.toString() + " - Health Potion";
    }
}
```

```
    Weapon weapon = new Weapon(1.3, 0.7, 100);
    HealthPotion potion = new HealthPotion(10.0, 0.0, 6);
    System.out.println(weapon);
    System.out.println(potion);
```



## Stack

| | Name | Value |
|---|---|---|
| | weapon | 0x350 |
| Weapon | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | damage | 100 |
| GameItem | this | 0x350 |
| | xLoc | 1.3 |
| | yLoc | 0.7 |
| | potion | 0x480 |
| HealthPotion | this | 0x480 |
| | xLoc | 10.0 |
| | yLoc | 0.0 |
| | increase | 6 |
| GameItem | this | 0x480 |
| | xLoc | 10.0 |
| | yLoc | 0.0 |
| toString | this | 0x350 |
| toString | this | 0x480 |
| toString | this | 0x480 |

## Heap

**Weapon**

| | |
|---|---|
| xLoc | 1.3 |
| yLoc | 0.7 |
| damage | 100 |

0x350

**HealthPotion**

| | |
|---|---|
| xLoc | 10.0 |
| yLoc | 0.0 |
| increase | 6 |

0x480

## in/out

```
Weapon Damage: 100
x: 10.0 y:0.0 - Health Potion
```

- Print the final String to the screen
- End program