Polymorphism

Polymorphism

If an object **is a** *type*

It can be stored in variables of that *type*

```java
public interface Usable {
    void use(Player player);
}
```

```java
public abstract class GameItem {
    private double loc;
    public GameItem(double loc) {this.loc = loc;}
}
```

```java
public class Weapon extends GameItem implements Usable {
    private int damage;
    public Weapon(double loc, int damage) {
        super(loc);
        this.damage = damage;
    }
    public int getDamage() {return damage;}
    @Override
    public void use(Player player) {
        player.setDamage(this.damage);
    }
}
```

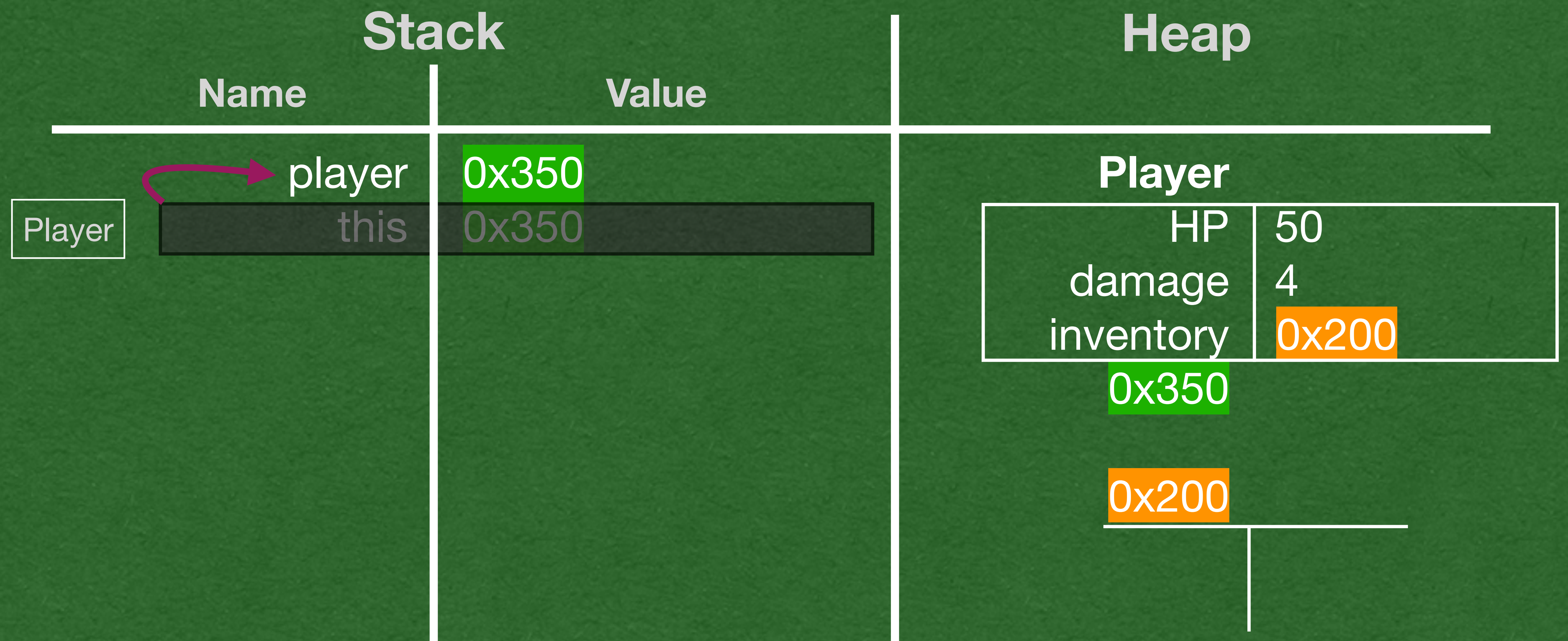
```java
public class Potion extends GameItem implements Usable {
   public Potion(double loc) {
        super(loc);
    }
    @Override
    public void use(Player player) {
        player.setHP(player.getHP() + 20);
    }
}
```

```java
public static void main(String[] args) {
    Player player = new Player();
    Weapon weapon = new Weapon(-5.0, 15);
    Usable potion = new Potion(3.5);
    player.pickUp(weapon);
    player.pickUp(potion);
    player.useAllItems();
}
```

```java
public class Player {
  private int HP = 50;
  private int damage = 4;
  private ArrayList<Usable> inventory=new ArrayList<>();
  public void pickUp(Usable item) {
    this.inventory.add(item);
  }
  public void useAllItems() {
    for (Usable item : this.inventory) {
      item.use(this);
    }
  }
  public int getHP() {return HP;}
  public void setHP(int HP) {this.HP = HP;}
  public void setDamage(int damage) {
    this.damage = damage;
  }
}
```

Memory Diagram

```
public class Player {
  private int HP = 50;
  private int damage = 4;
  private ArrayList<Usable> inventory=new ArrayList<>();
  public void pickUp(Usable item) {
    this.inventory.add(item);
  }
  public void useAllItems() {
    for (Usable item : this.inventory) {
      item.use(this);
    }
  }
  public int getHP() {return HP;}
  public void setHP(int HP) {this.HP = HP;}
  public void setDamage(int damage) {
    this.damage = damage;
  }
}
```



Stack

Name	Value
player	0x350
this	0x350

Heap

Player

HP	50
damage	4
inventory	0x200

0x350

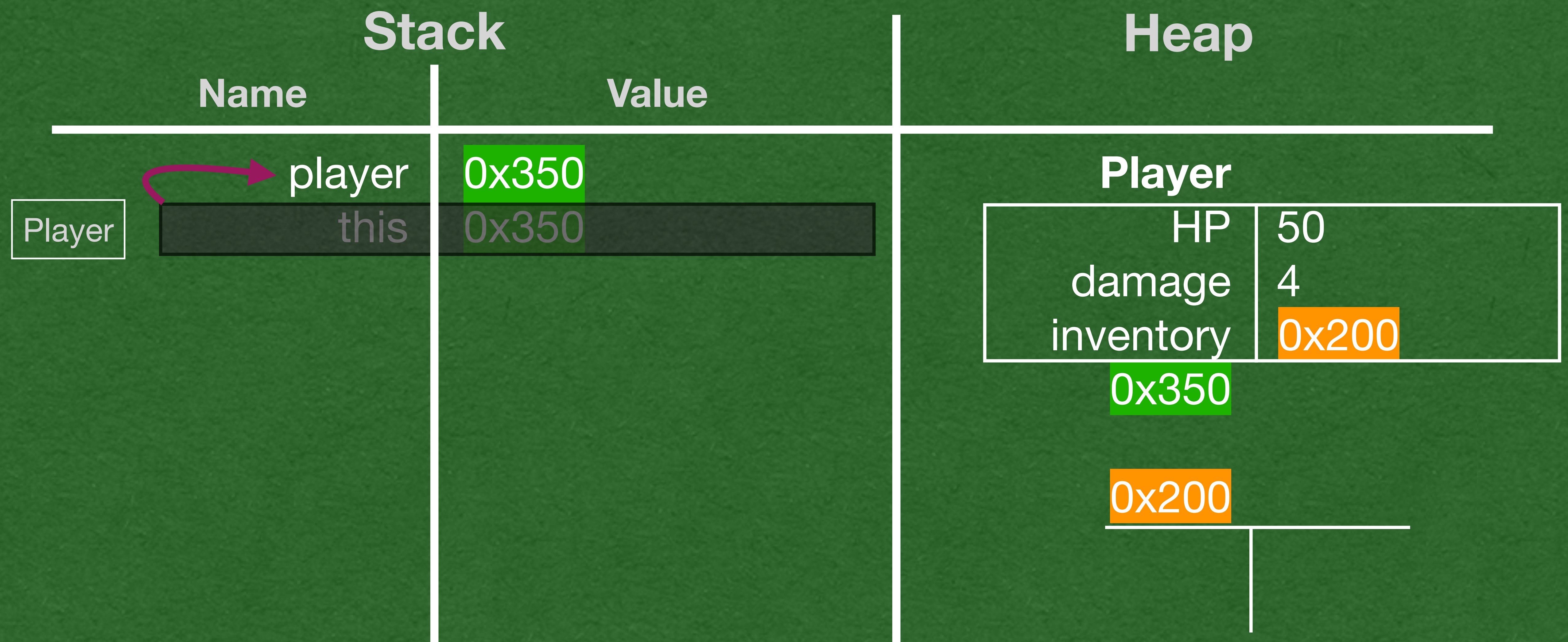
0x200

```
public static void main(String[] args) {
   Player player = new Player();
   Weapon weapon = new Weapon(-5.0, 15);
   Usable potion = new Potion(3.5);
   player.pickUp(weapon);
   player.pickUp(potion);
   player.useAllItems();
}
```

in/out

- Construct the Player object
- Default constructor is called since no constructor is explicitly defined

```
public class Player {
  private int HP = 50;
  private int damage = 4;
  private ArrayList<Usable> inventory=new ArrayList<>();
  public void pickUp(Usable item) {
    this.inventory.add(item);
  }
  public void useAllItems() {
    for (Usable item : this.inventory) {
      item.use(this);
    }
  }
  public int getHP() {return HP;}
  public void setHP(int HP) {this.HP = HP;}
  public void setDamage(int damage) {
    this.damage = damage;
  }
}
```



in/out

- Player creates a new ArrayList while initializing variables

```
public static void main(String[] args) {
  Player player = new Player();
  Weapon weapon = new Weapon(-5.0, 15);
  Usable potion = new Potion(3.5);
  player.pickUp(weapon);
  player.pickUp(potion);
  player.useAllItems();
}
```

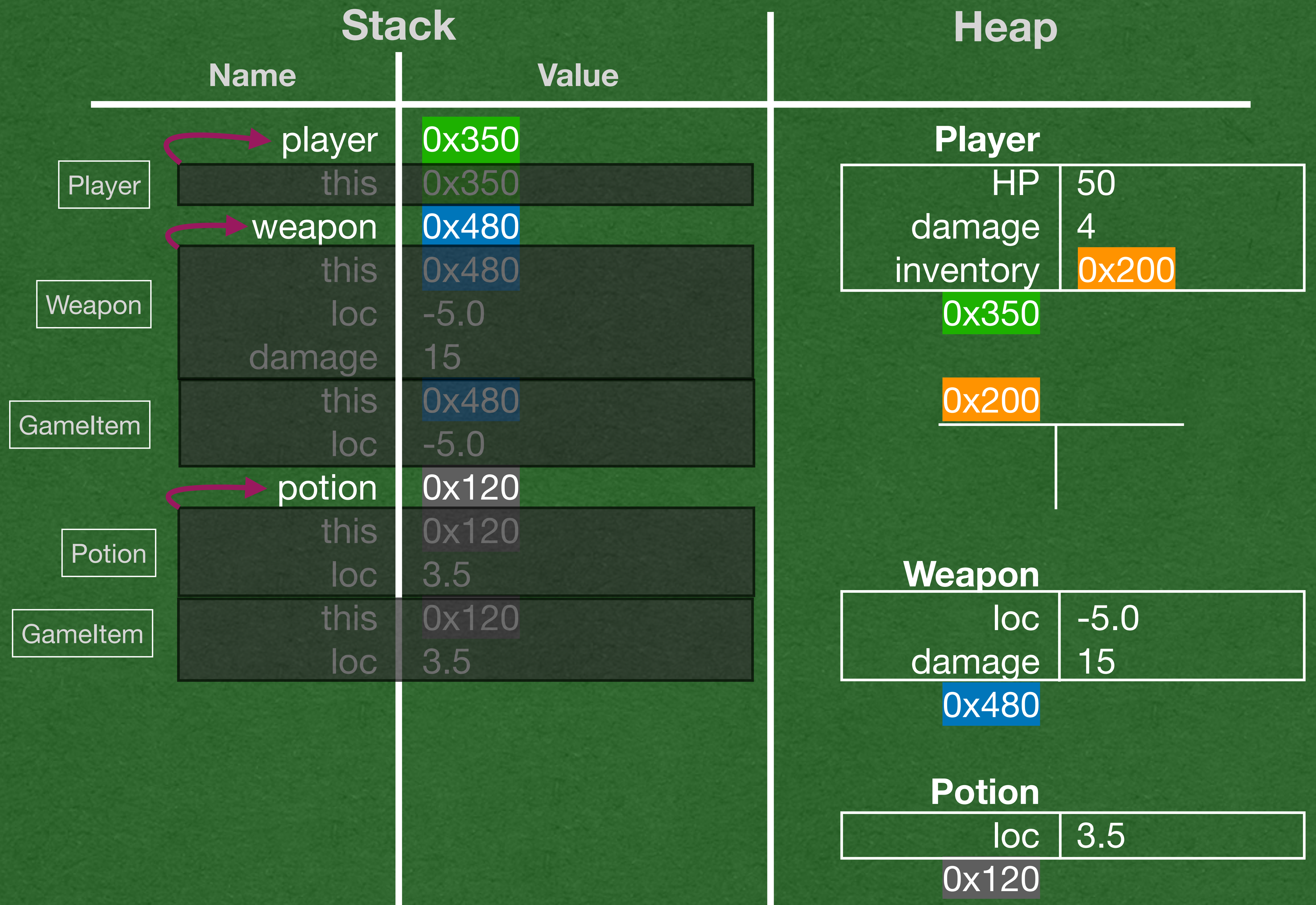
```java
public interface Usable {
    void use(Player player);
}
```

```java
public abstract class GameItem {
    private double loc;
    public GameItem(double loc) {this.loc = loc;}
}
```

```java
public class Weapon extends GameItem implements Usable {
    private int damage;
    public Weapon(double loc, int damage) {
        super(loc);
        this.damage = damage;
    }
    public int getDamage() {return damage;}
    @Override
    public void use(Player player) {
        player.setDamage(this.damage);
    }
}
```

```java
public class Potion extends GameItem implements Usable {
   public Potion(double loc) {
        super(loc);
    }
    @Override
    public void use(Player player) {
        player.setHP(player.getHP() + 20);
    }
}
```

```java
public static void main(String[] args) {
    Player player = new Player();
    Weapon weapon = new Weapon(-5.0, 15);
    Usable potion = new Potion(3.5);
    player.pickUp(weapon);
    player.pickUp(potion);
    player.useAllItems();
}
```



Stack

	Name	Value
	player	0x350
Player	this	0x350
	weapon	0x480
Weapon	this	0x480
	loc	-5.0
	damage	15
GameItem	this	0x480
	loc	-5.0
	potion	0x120
Potion	this	0x120
	loc	3.5
GameItem	this	0x120
	loc	3.5

Heap

Player

HP	50
damage	4
inventory	0x200

0x350

0x200

Weapon

loc	-5.0
damage	15

0x480

Potion

loc	3.5

0x120

in/out

- Construct the Weapon and Potion objects
- Remember to call the super constructor

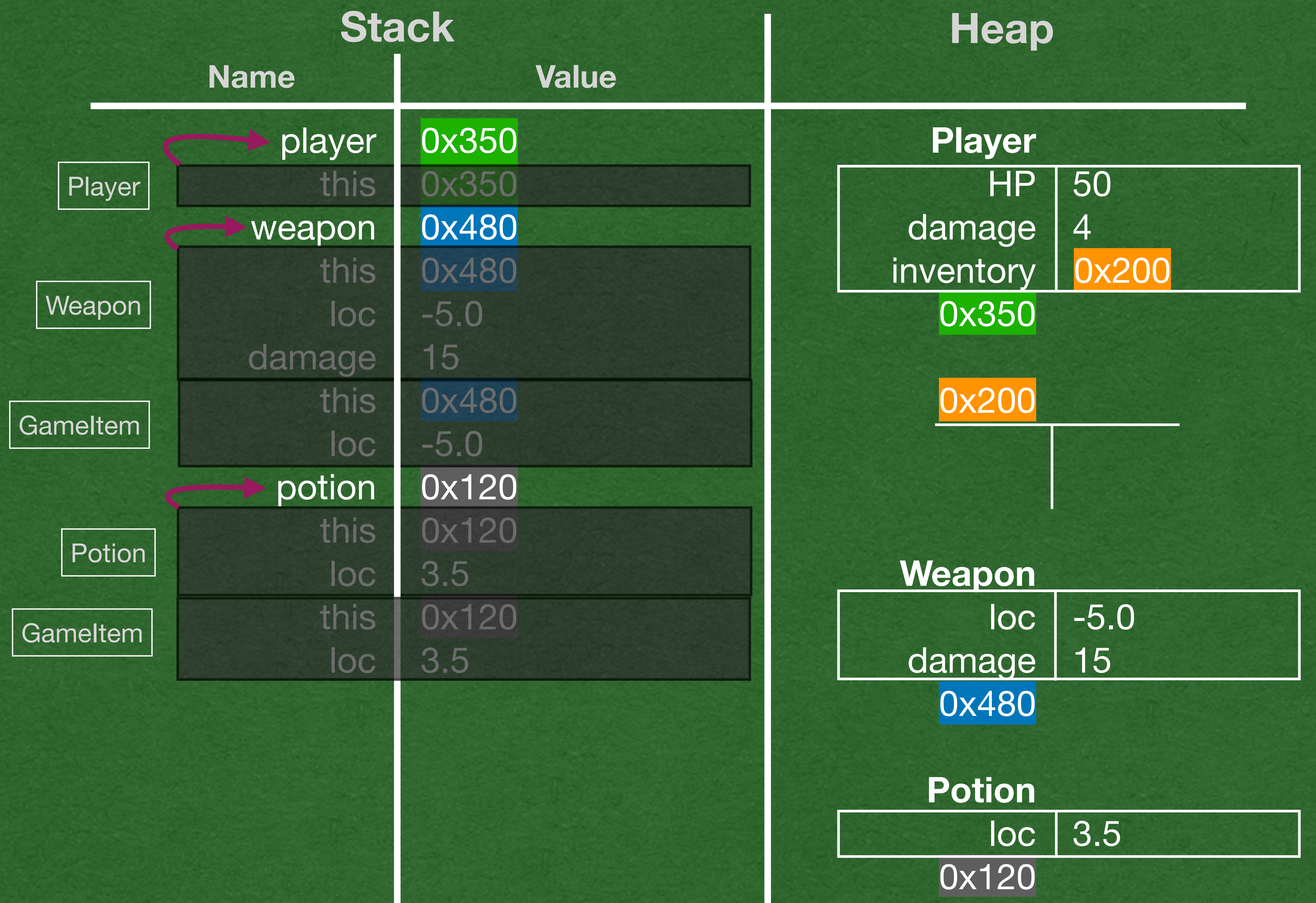
```
public interface Usable {
    void use(Player player);
}
```

```
public abstract class GameItem {
    private double loc;
    public GameItem(double loc) {this.loc = loc;}
}
```

```
public class Weapon extends GameItem implements Usable {
    private int damage;
    public Weapon(double loc, int damage) {
        super(loc);
        this.damage = damage;
    }
    public int getDamage() {return damage;}
    @Override
    public void use(Player player) {
        player.setDamage(this.damage);
    }
}
```

```
public class Potion extends GameItem implements Usable {
   public Potion(double loc) {
        super(loc);
    }
    @Override
    public void use(Player player) {
        player.setHP(player.getHP() + 20);
    }
}
```

```
public static void main(String[] args) {
    Player player = new Player();
    Weapon weapon = new Weapon(-5.0, 15);
    Usable potion = new Potion(3.5);
    player.pickUp(weapon);
    player.pickUp(potion);
    player.useAllItems();
}
```



Stack

	Name	Value
	player	0x350
Player	this	0x350
	weapon	0x480
Weapon	this	0x480
	loc	-5.0
	damage	15
GameItem	this	0x480
	loc	-5.0
	potion	0x120
Potion	this	0x120
	loc	3.5
GameItem	this	0x120
	loc	3.5

Heap

Player

HP	50
damage	4
inventory	0x200

0x350

0x200

Weapon

loc	-5.0
damage	15

0x480

Potion

loc	3.5

0x120

in/out

- Nothing on the stack for the interface
- Interfaces do not have constructors
    - Not even the default constructor

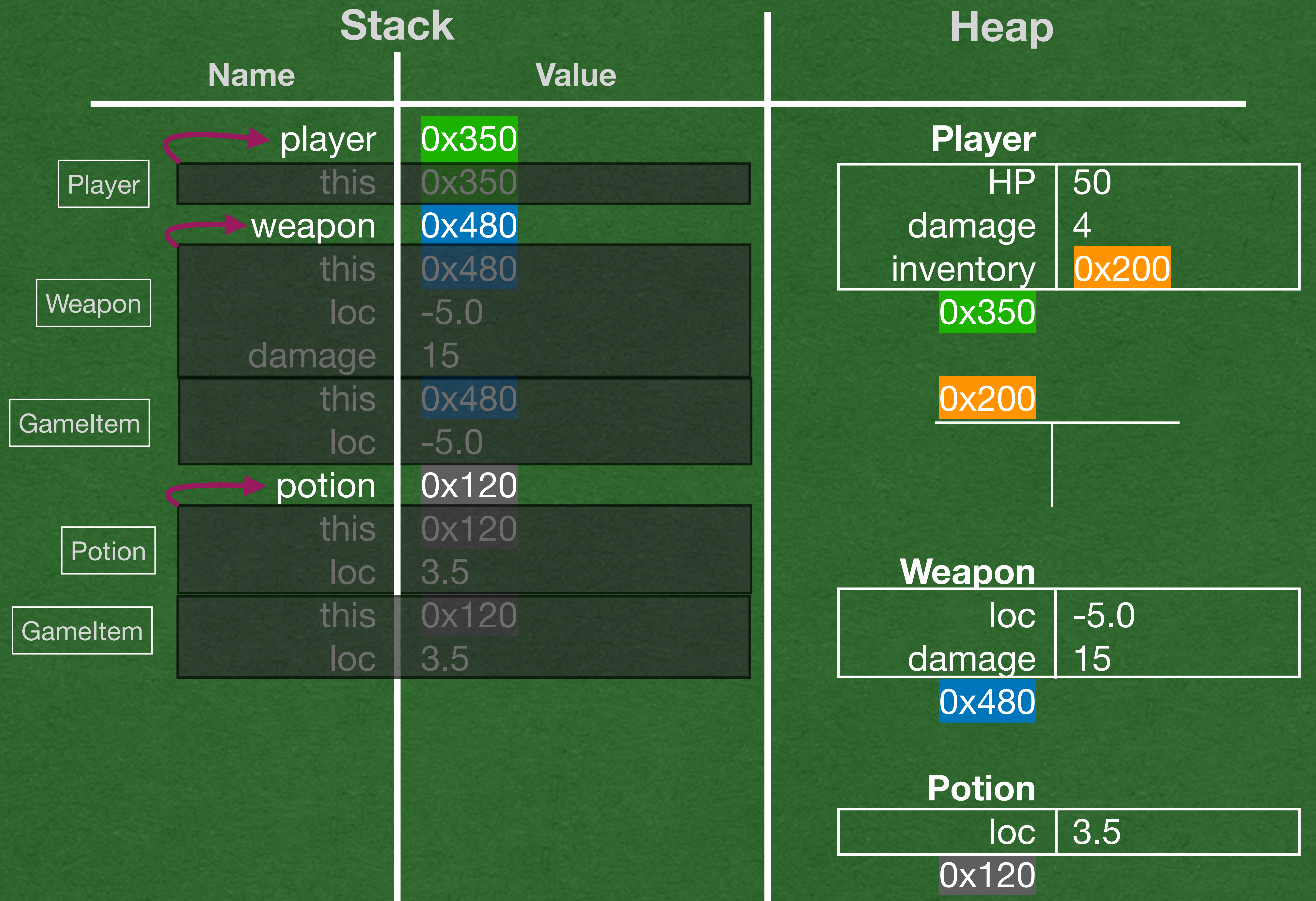
```java
public interface Usable {
    void use(Player player);
}
```

```java
public abstract class GameItem {
    private double loc;
    public GameItem(double loc) {this.loc = loc;}
}
```

```java
public class Weapon extends GameItem implements Usable {
    private int damage;
    public Weapon(double loc, int damage) {
        super(loc);
        this.damage = damage;
    }
    public int getDamage() {return damage;}
    @Override
    public void use(Player player) {
        player.setDamage(this.damage);
    }
}
```

```java
public class Potion extends GameItem implements Usable {
   public Potion(double loc) {
        super(loc);
    }
    @Override
    public void use(Player player) {
        player.setHP(player.getHP() + 20);
    }
}
```

```java
public static void main(String[] args) {
    Player player = new Player();
    Weapon weapon = new Weapon(-5.0, 15);
    Usable potion = new Potion(3.5);
    player.pickUp(weapon);
    player.pickUp(potion);
    player.useAllItems();
}
```



Stack

	Name	Value
	player	0x350
Player	this	0x350
	weapon	0x480
Weapon	this	0x480
	loc	-5.0
	damage	15
GameItem	this	0x480
	loc	-5.0
	potion	0x120
Potion	this	0x120
	loc	3.5
GameItem	this	0x120
	loc	3.5

Heap

Player

HP	50
damage	4
inventory	0x200

0x350

0x200

Weapon

loc	-5.0
damage	15

0x480

Potion

loc	3.5

0x120

in/out

- The Weapon is stored in a Weapon variable
- Can call every method known to the Weapon class

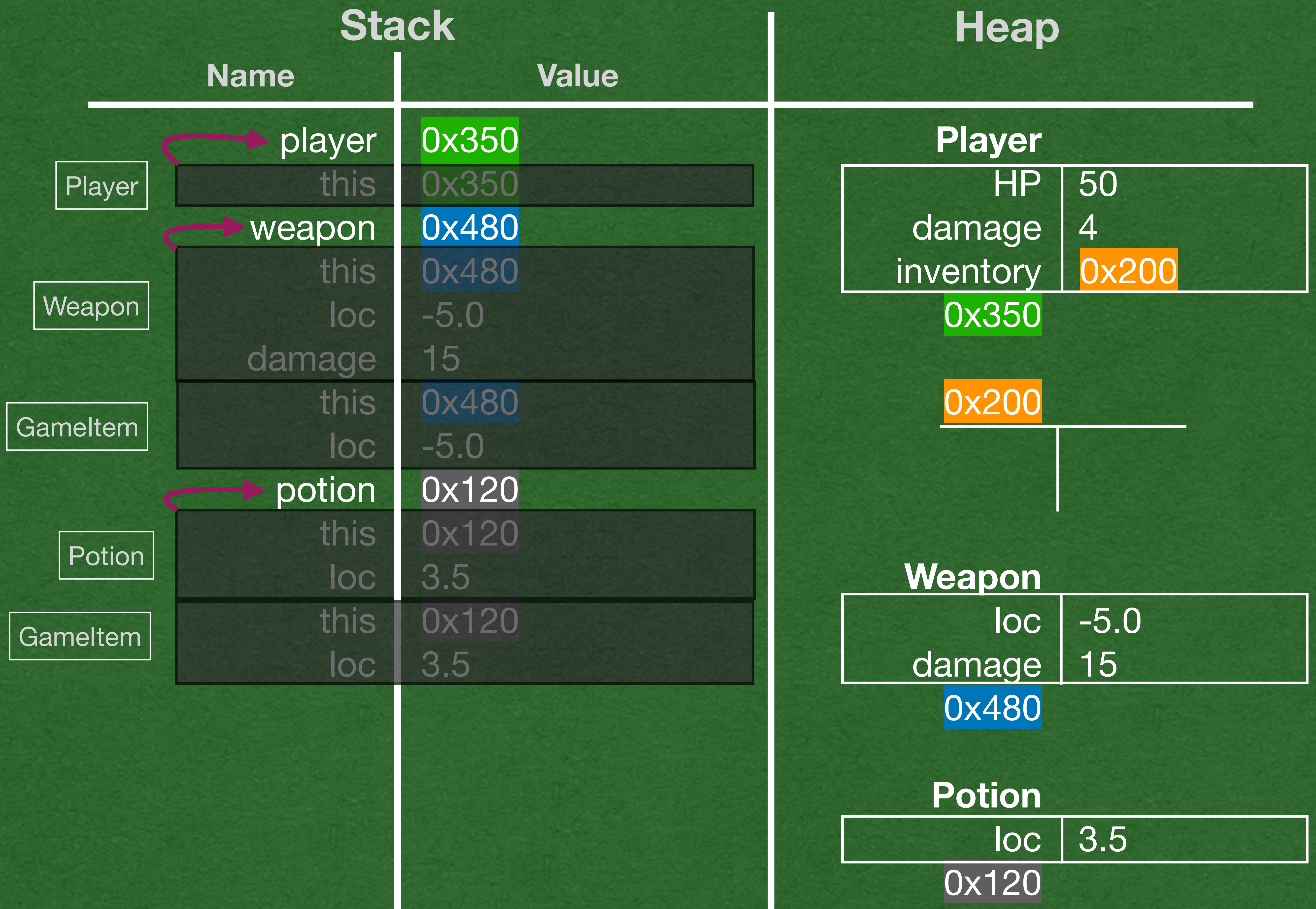
```java
public interface Usable {
    void use(Player player);
}
```

```java
public abstract class GameItem {
    private double loc;
    public GameItem(double loc) {this.loc = loc;}
}
```

```java
public class Weapon extends GameItem implements Usable {
    private int damage;
    public Weapon(double loc, int damage) {
        super(loc);
        this.damage = damage;
    }
    public int getDamage() {return damage;}
    @Override
    public void use(Player player) {
        player.setDamage(this.damage);
    }
}
```

```java
public class Potion extends GameItem implements Usable {
   public Potion(double loc) {
        super(loc);
    }
    @Override
    public void use(Player player) {
        player.setHP(player.getHP() + 20);
    }
}
```

```java
public static void main(String[] args) {
    Player player = new Player();
    Weapon weapon = new Weapon(-5.0, 15);
    Usable potion = new Potion(3.5);
    player.pickUp(weapon);
    player.pickUp(potion);
    player.useAllItems();
}
```



Stack

	Name	Value
	player	0x350
Player	this	0x350
	weapon	0x480
Weapon	this	0x480
	loc	-5.0
	damage	15
GameItem	this	0x480
	loc	-5.0
	potion	0x120
Potion	this	0x120
	loc	3.5
GameItem	this	0x120
	loc	3.5

Heap

Player

HP	50
damage	4
inventory	0x200

0x350

0x200

Weapon

loc	-5.0
damage	15

0x480

Potion

loc	3.5

0x120

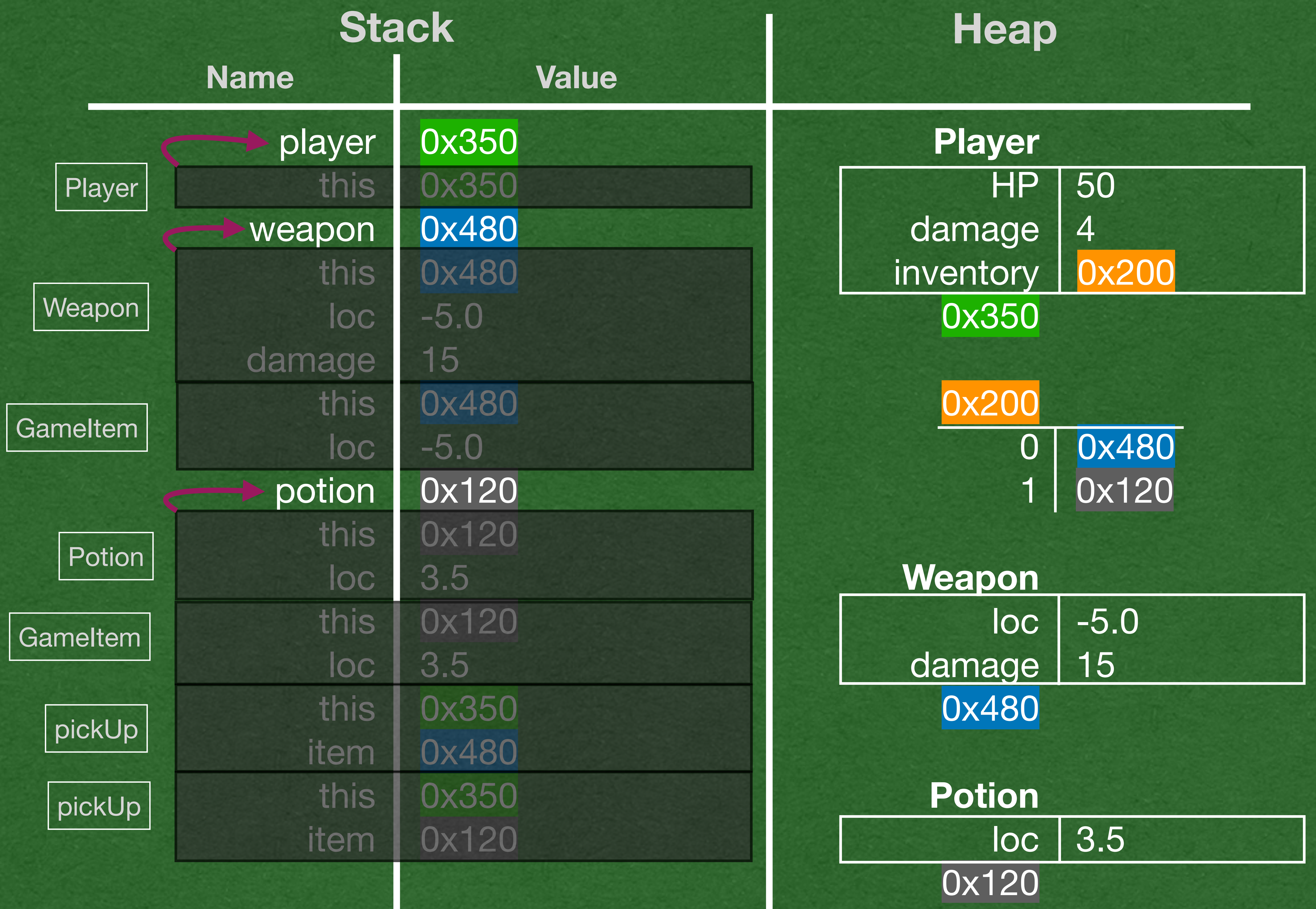
in/out

- The Potion is stored in a Usable variable
- Can only call the use method from this variable

```
public class Player {
  private int HP = 50;
  private int damage = 4;
  private ArrayList<Usable> inventory=new ArrayList<>();
  public void pickUp(Usable item) {
    this.inventory.add(item);
  }
  public void useAllItems() {
    for (Usable item : this.inventory) {
      item.use(this);
    }
  }
  public int getHP() {return HP;}
  public void setHP(int HP) {this.HP = HP;}
  public void setDamage(int damage) {
    this.damage = damage;
  }
}
```

```
public interface Usable {
    void use(Player player);
}
```

```
public static void main(String[] args) {
    Player player = new Player();
    Weapon weapon = new Weapon(-5.0, 15);
    Usable potion = new Potion(3.5);
    player.pickUp(weapon);
    player.pickUp(potion);
    player.useAllItems();
}
```



Stack

	Name	Value
	player	0x350
Player	this	0x350
	weapon	0x480
Weapon	this	0x480
	loc	-5.0
	damage	15
GameItem	this	0x480
	loc	-5.0
	potion	0x120
Potion	this	0x120
	loc	3.5
GameItem	this	0x120
	loc	3.5
pickUp	this	0x350
	item	0x480
pickUp	this	0x350
	item	0x120

Heap

Player

HP	50
damage	4
inventory	0x200

0x350

0x200

0	0x480
1	0x120

Weapon

loc	-5.0
damage	15

0x480

Potion

loc	3.5

0x120

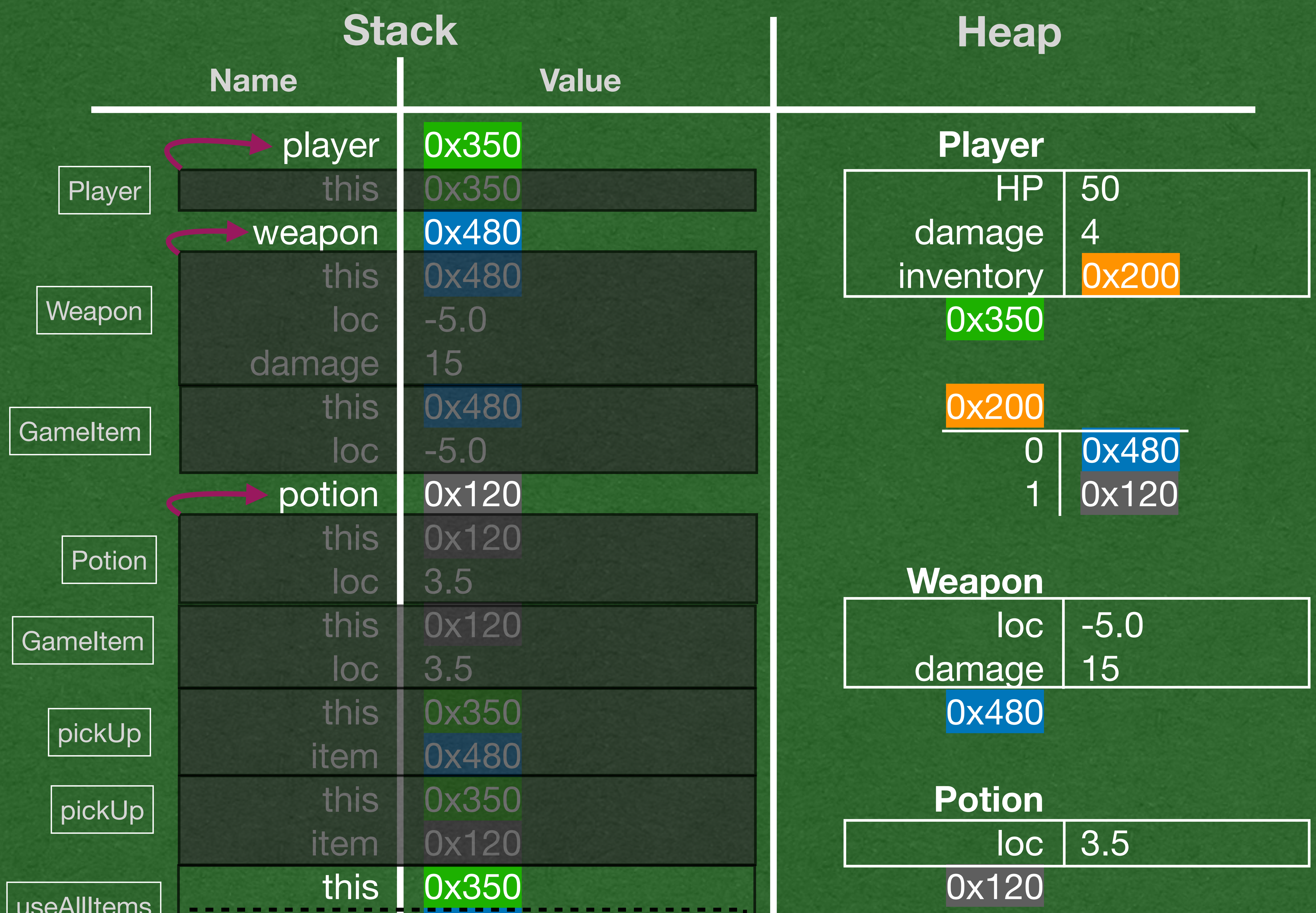
in/out

- pickUp takes a Usable
- Weapon and Potion both implement Usable so they can be picked up

```java
public class Player {
  private int HP = 50;
  private int damage = 4;
  private ArrayList<Usable> inventory=new ArrayList<>();
  public void pickUp(Usable item) {
    this.inventory.add(item);
  }
  public void useAllItems() {
    for (Usable item : this.inventory) {
      item.use(this);
    }
  }
  public int getHP() {return HP;}
  public void setHP(int HP) {this.HP = HP;}
  public void setDamage(int damage) {
    this.damage = damage;
  }
}
```

```java
public interface Usable {
    void use(Player player);
}
```

```java
public static void main(String[] args) {
    Player player = new Player();
    Weapon weapon = new Weapon(-5.0, 15);
    Usable potion = new Potion(3.5);
    player.pickUp(weapon);
    player.pickUp(potion);
    player.useAllItems();
}
```



Stack

	Name	Value
	player	0x350
Player	this	0x350
	weapon	0x480
Weapon	this	0x480
	loc	-5.0
	damage	15
GameItem	this	0x480
	loc	-5.0
	potion	0x120
Potion	this	0x120
	loc	3.5
GameItem	this	0x120
	loc	3.5
pickUp	this	0x350
	item	0x480
pickUp	this	0x350
	item	0x120
useAllItems	this	0x350
	item	0x480

Heap

Player (0x350)

HP	50
damage	4
inventory	0x200

0x200

0	0x480
1	0x120

Weapon (0x480)

loc	-5.0
damage	15

Potion (0x120)

loc	3.5

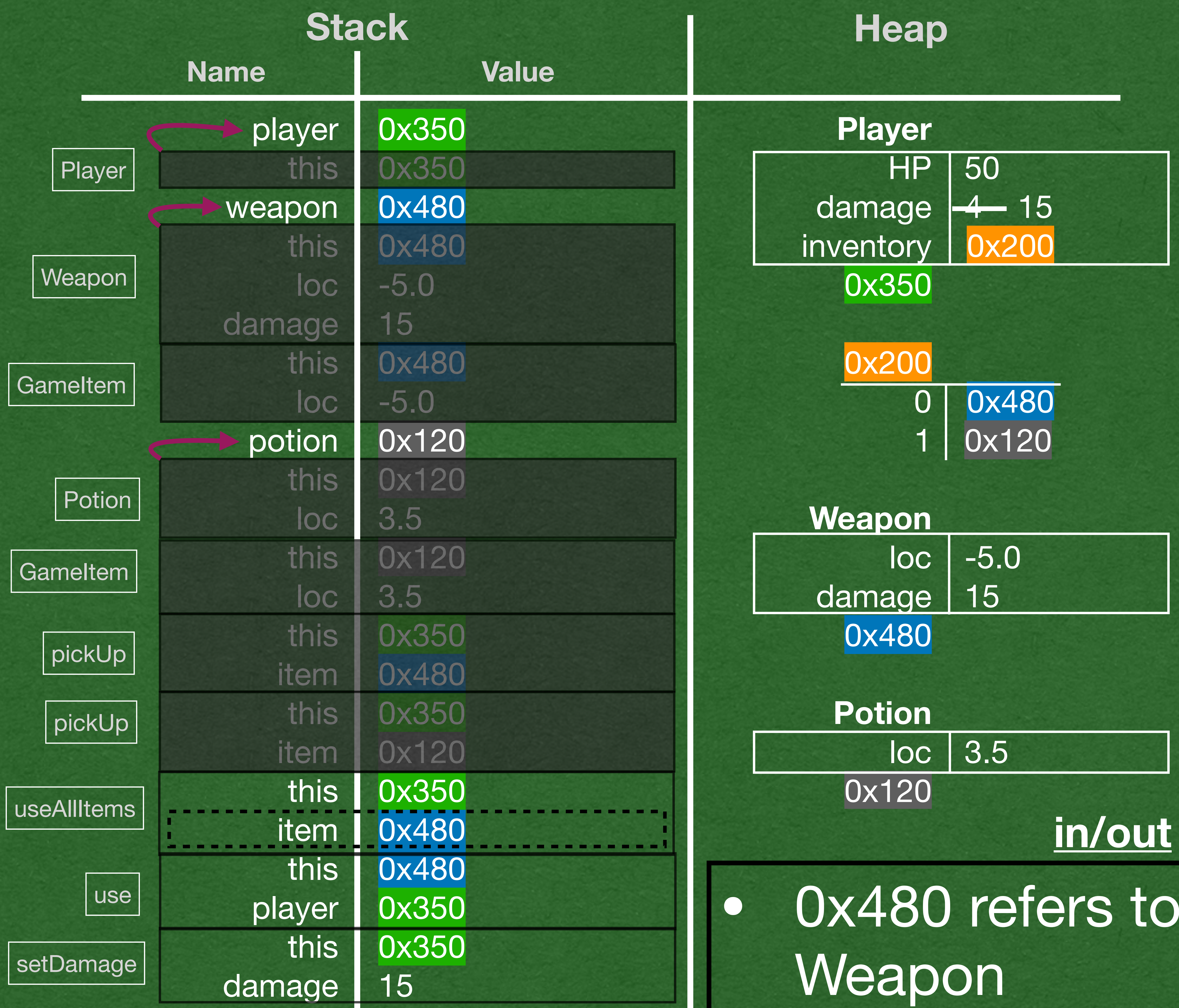
in/out

- useAllItems loops through all Usables
- Can only call use from the item variable

```
public class Player {
  private int HP = 50;
  private int damage = 4;
  private ArrayList<Usable> inventory=new ArrayList<>();
  public void pickUp(Usable item) {
    this.inventory.add(item);
  }
  public void useAllItems() {
    for (Usable item : this.inventory) {
      item.use(this);
    }
  }
  public int getHP() {return HP;}
  public void setHP(int HP) {this.HP = HP;}
  public void setDamage(int damage) {
    this.damage = damage;
  }
}
```

```
public class Weapon extends GameItem implements Usable {
    private int damage;
    public Weapon(double loc, int damage) {
        super(loc);
        this.damage = damage;
    }
    public int getDamage() {return damage;}
    @Override
    public void use(Player player) {
        player.setDamage(this.damage);
    }
}
```

```
public static void main(String[] args) {
    Player player = new Player();
    Weapon weapon = new Weapon(-5.0, 15);
    Usable potion = new Potion(3.5);
    player.pickUp(weapon);
    player.pickUp(potion);
    player.useAllItems();
}
```



Stack

	Name	Value
	player	0x350
Player	this	0x350
	weapon	0x480
Weapon	this	0x480
	loc	-5.0
	damage	15
GameItem	this	0x480
	loc	-5.0
	potion	0x120
Potion	this	0x120
	loc	3.5
GameItem	this	0x120
	loc	3.5
pickUp	this	0x350
	item	0x480
pickUp	this	0x350
	item	0x120
useAllItems	this	0x350
	item	0x480
use	this	0x480
	player	0x350
setDamage	this	0x350
	damage	15

Heap

Player

HP	50
damage	~~4~~ 15
inventory	0x200

0x350

0x200

0	0x480
1	0x120

Weapon

loc	-5.0
damage	15

0x480

Potion

loc	3.5

0x120

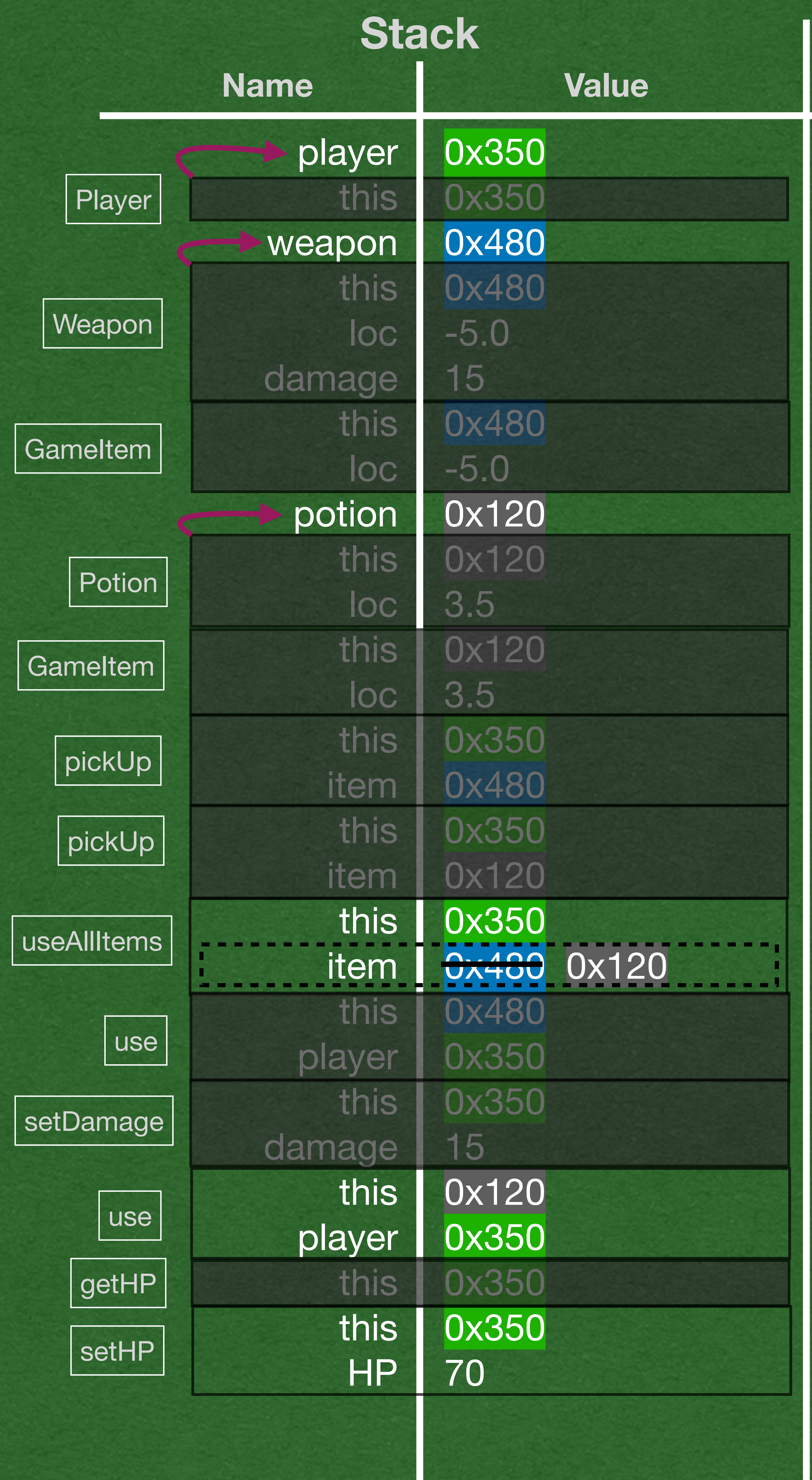
in/out

- 0x480 refers to a Weapon
- The use method in the Weapon class is called

```java
public class Player {
  private int HP = 50;
  private int damage = 4;
  private ArrayList<Usable> inventory=new ArrayList<>();
  public void pickUp(Usable item) {
    this.inventory.add(item);
  }
  public void useAllItems() {
    for (Usable item : this.inventory) {
      item.use(this);
    }
  }
  public int getHP() {return HP;}
  public void setHP(int HP) {this.HP = HP;}
  public void setDamage(int damage) {
    this.damage = damage;
  }
}
```

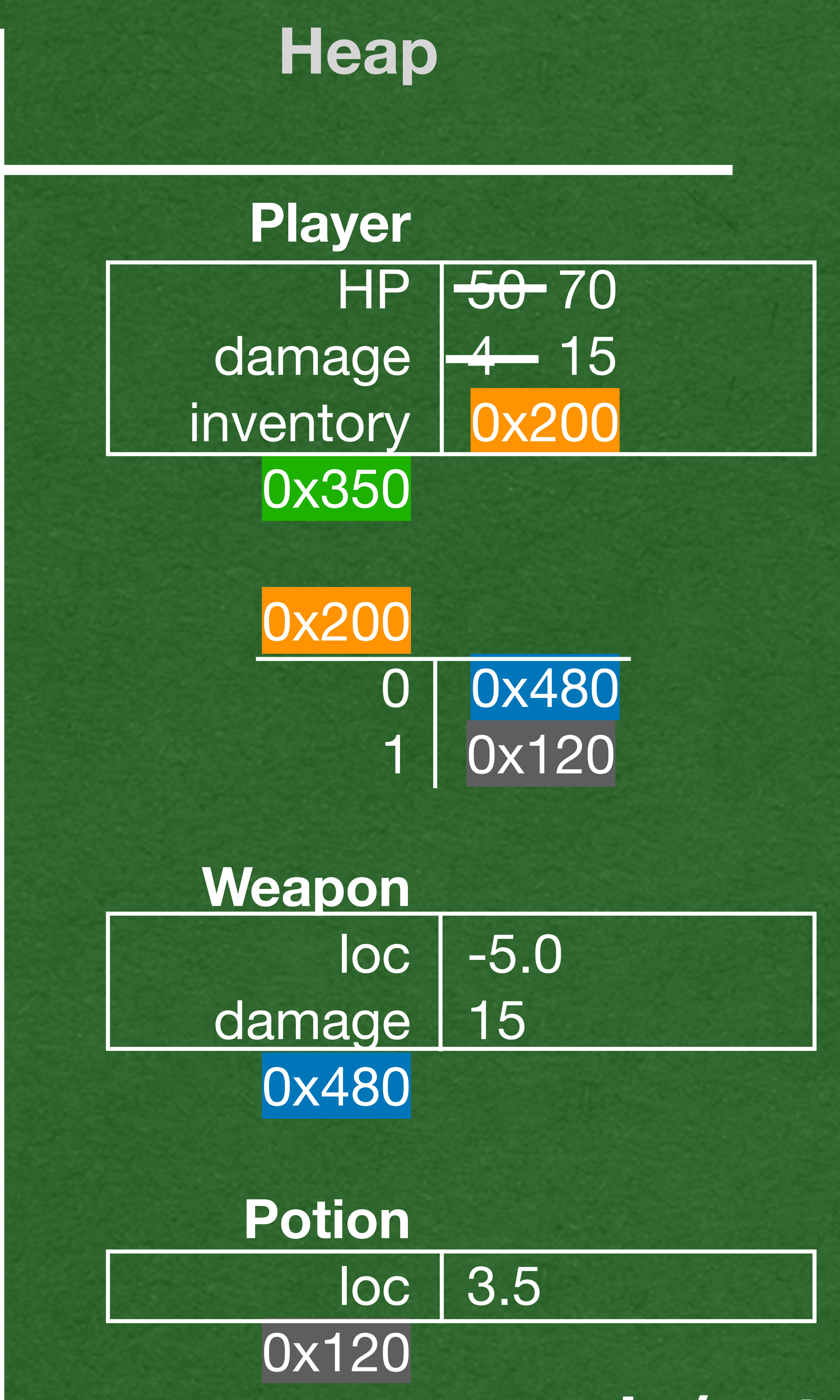
```java
public class Potion extends GameItem implements Usable {
   public Potion(double loc) {
       super(loc);
   }
   @Override
   public void use(Player player) {
       player.setHP(player.getHP() + 20);
   }
}
```

```java
public static void main(String[] args) {
    Player player = new Player();
    Weapon weapon = new Weapon(-5.0, 15);
    Usable potion = new Potion(3.5);
    player.pickUp(weapon);
    player.pickUp(potion);
    player.useAllItems();
}
```



Stack

	Name	Value
	player	0x350
Player	this	0x350
	weapon	0x480
Weapon	this	0x480
	loc	-5.0
	damage	15
GameItem	this	0x480
	loc	-5.0
	potion	0x120
Potion	this	0x120
	loc	3.5
GameItem	this	0x120
	loc	3.5
pickUp	this	0x350
	item	0x480
pickUp	this	0x350
	item	0x120
useAllItems	this	0x350
	item	~~0x480~~ 0x120
use	this	0x480
	player	0x350
setDamage	this	0x350
	damage	15
use	this	0x120
	player	0x350
getHP	this	0x350
setHP	this	0x350
	HP	70



Heap

Player

HP	~~50~~ 70
damage	~~4~~ 15
inventory	0x200

0x350

0x200

0	0x480
1	0x120

Weapon

loc	-5.0
damage	15

0x480

Potion

loc	3.5

0x120

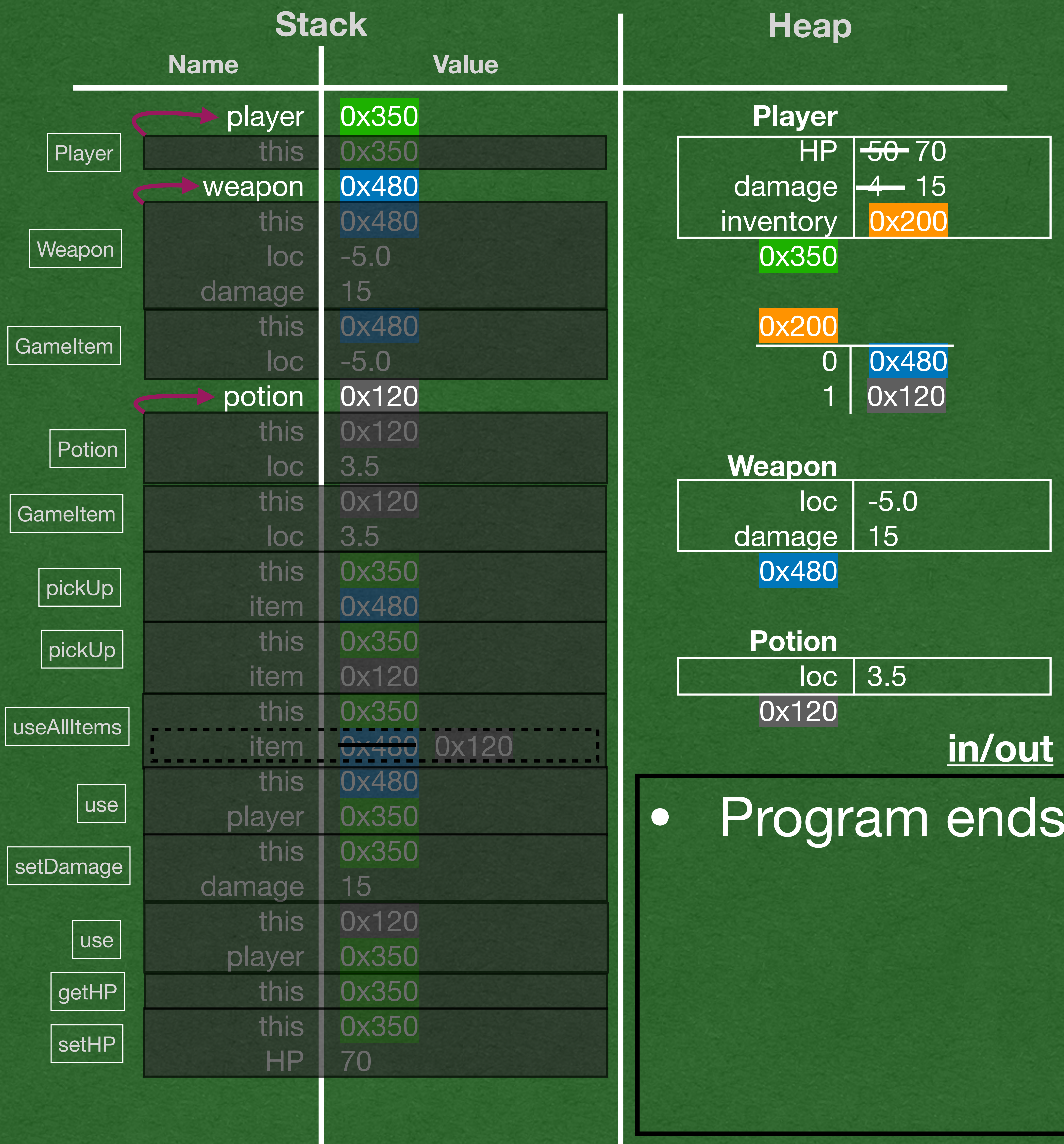
in/out

- 0x120 refers to a Potion
- The use method in the Potion class is called

```java
public class Player {
  private int HP = 50;
  private int damage = 4;
  private ArrayList<Usable> inventory=new ArrayList<>();
  public void pickUp(Usable item) {
    this.inventory.add(item);
  }
  public void useAllItems() {
    for (Usable item : this.inventory) {
      item.use(this);
    }
  }
  public int getHP() {return HP;}
  public void setHP(int HP) {this.HP = HP;}
  public void setDamage(int damage) {
    this.damage = damage;
  }
}
```

```java
public class Potion extends GameItem implements Usable {
   public Potion(double loc) {
        super(loc);
    }
    @Override
    public void use(Player player) {
        player.setHP(player.getHP() + 20);
    }
}
```

```java
public static void main(String[] args) {
    Player player = new Player();
    Weapon weapon = new Weapon(-5.0, 15);
    Usable potion = new Potion(3.5);
    player.pickUp(weapon);
    player.pickUp(potion);
    player.useAllItems();
```



Stack

	Name	Value
	player	0x350
Player	this	0x350
	weapon	0x480
Weapon	this	0x480
	loc	-5.0
	damage	15
GameItem	this	0x480
	loc	-5.0
	potion	0x120
Potion	this	0x120
	loc	3.5
GameItem	this	0x120
	loc	3.5
pickUp	this	0x350
	item	0x480
pickUp	this	0x350
	item	0x120
useAllItems	this	0x350
	item	~~0x480~~ 0x120
use	this	0x480
	player	0x350
setDamage	this	0x350
	damage	15
use	this	0x120
	player	0x350
getHP	this	0x350
setHP	this	0x350
	HP	70

Heap

Player

HP	~~50~~ 70
damage	~~4~~ 15
inventory	0x200

0x350

0x200

0	0x480
1	0x120

Weapon

loc	-5.0
damage	15

0x480

Potion

loc	3.5

0x120

in/out

- Program ends