Classes: Part 2

Memory Diagram

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

Stack

Name	Value

Heap

in/out

- We'll trace this version of the code
- Set up the stack, heap, and in/out

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

Stack

Name	Value

Heap

in/out

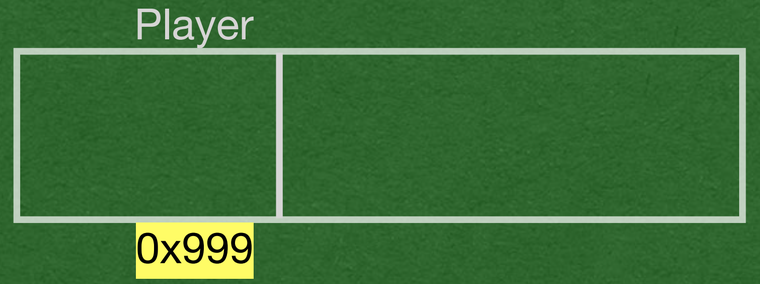
- New objects are created on the heap
- Only a reference is stored in variables

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

Stack

Name	Value

Heap



in/out

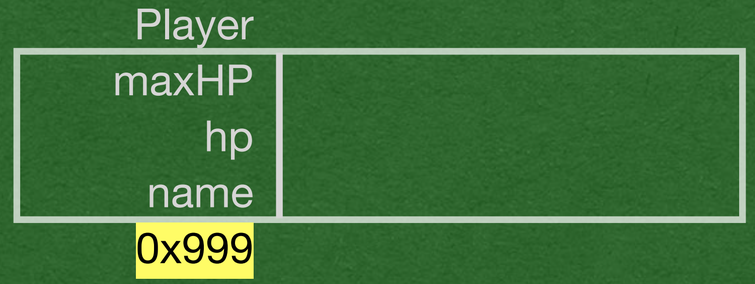
- Add the type of the object above it in the heap
- Make up a reference number and add it below the object

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

Stack

Name	Value

Heap



in/out

- Create all the instance variables and add them to the object
- Each new object will have its own copies of each instance variable

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

Stack

Name	Value
p1	
Player	
this	0x999
name	"Dark Cecil"
maxHP	10

Heap

Player

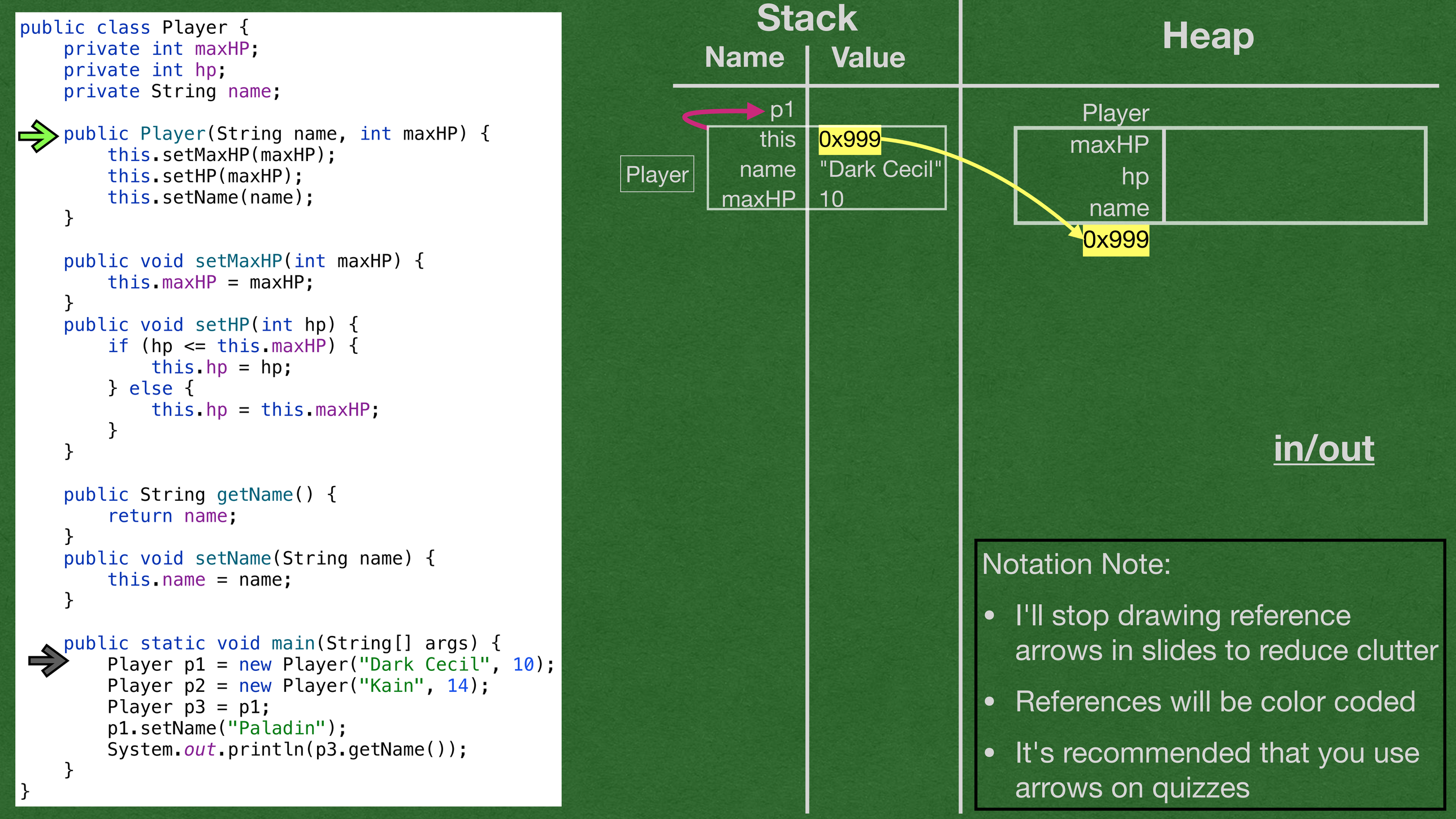
maxHP	
hp	
name	

0x999

in/out

- The constructor is a method
- Add it to the stack along with all of its parameters
- Constructors return a reference to the object that was created

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```



```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

Stack

Name	Value
p1	
Player	
this	0x999
name	"Dark Cecil"
maxHP	10

Heap

Player

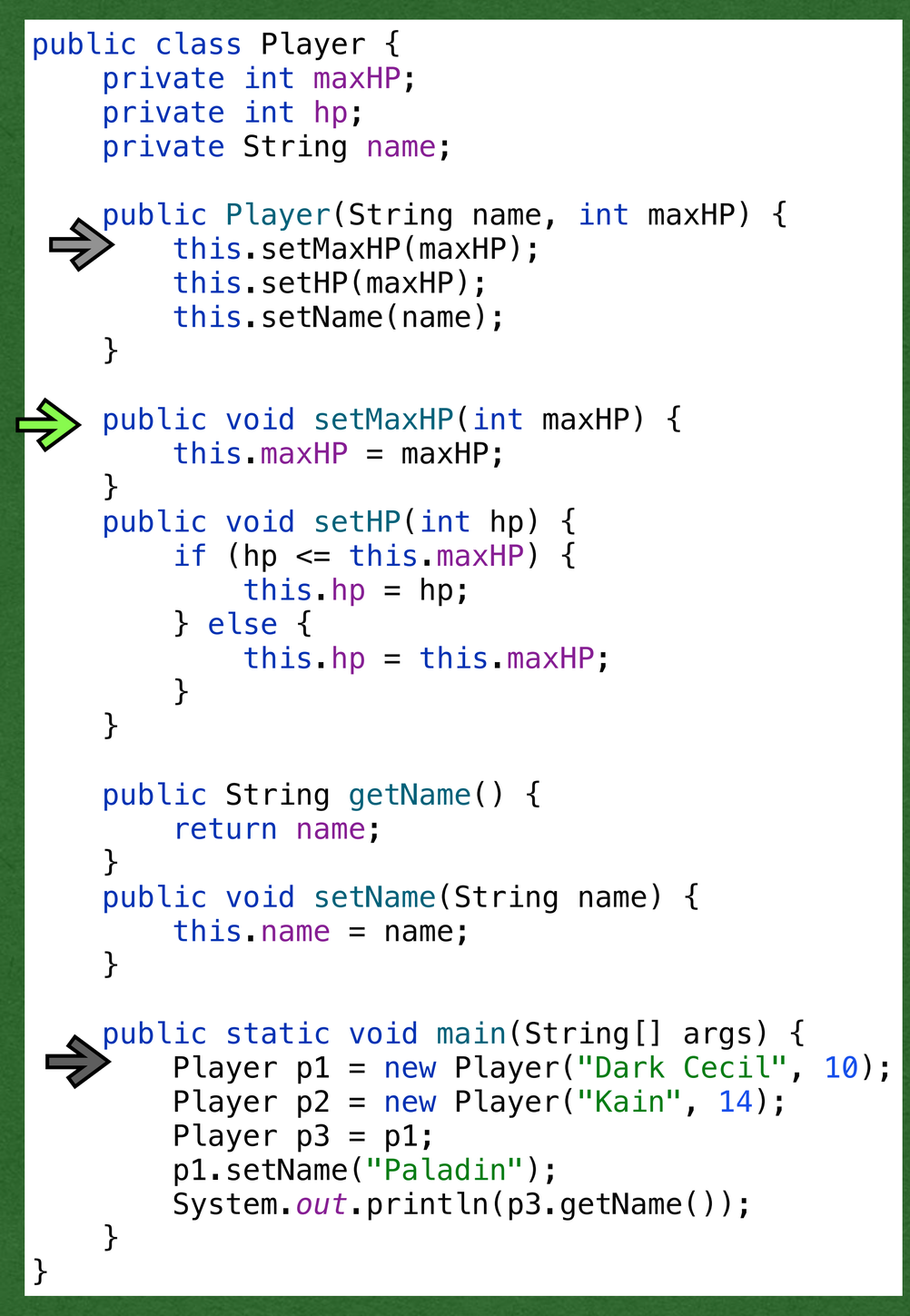
maxHP	
hp	
name	

0x999

in/out

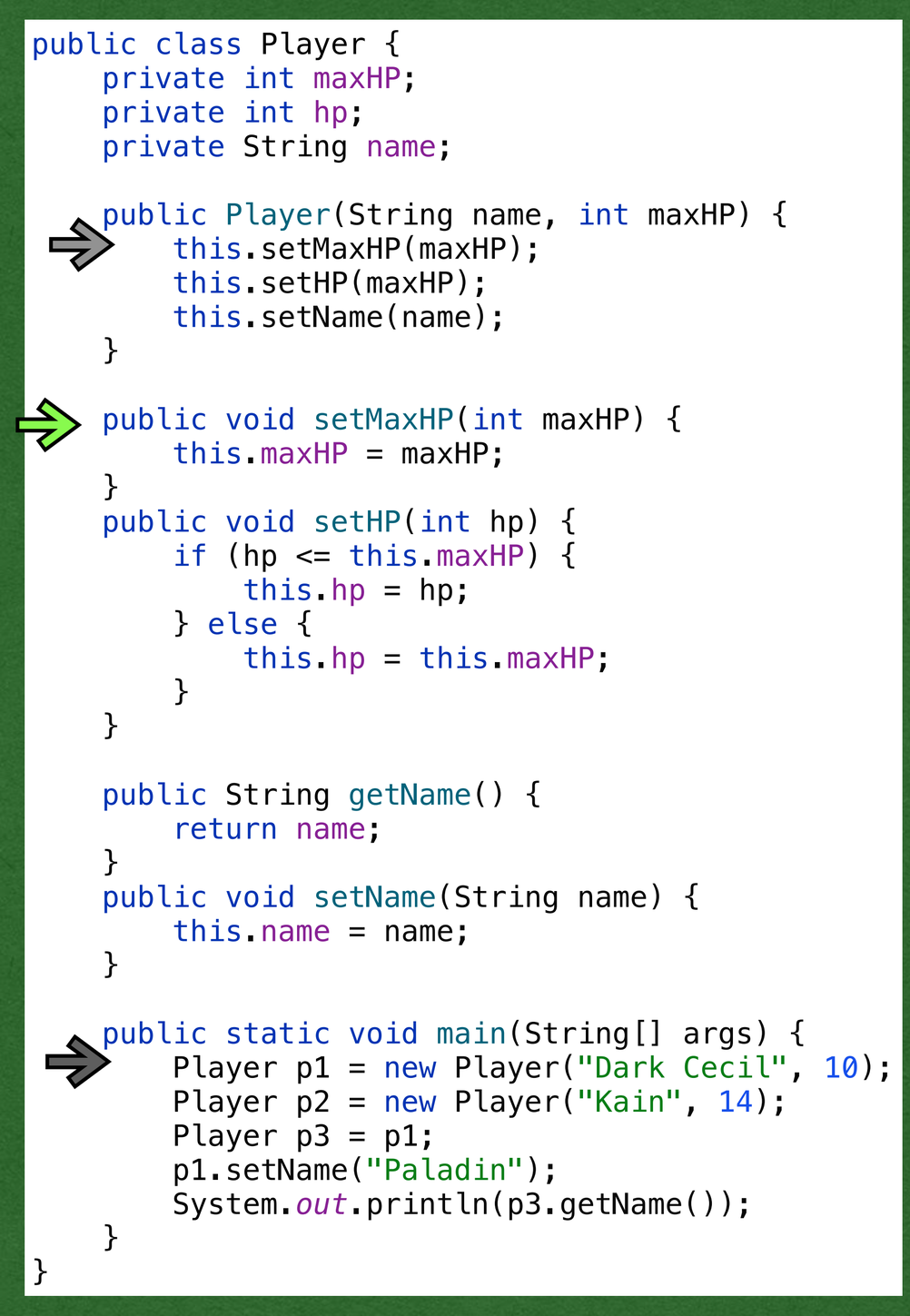
- All [non-static] methods have a "hidden" parameter of **this**
- For constructors, **this** stores a reference to the object that is being created

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```



- Our constructor calls a setter method
- Methods contain a reference a reference to the calling object in **this**

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```



in/out

- This method was called by **this** in the Player constructor stack frame which stores the reference 0x999
- 0x999 is the object that called setMaxHP so that stack frame's **this** stores 0x999

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

Stack

Name	Value
p1	
Player	
this	0x999
name	"Dark Cecil"
maxHP	10
setMaxHP	
this	0x999
maxHP	10

Heap

Player

maxHP	10
hp	
name	

0x999

in/out

- The setter method changes the value of a variable stored in the heap
- Follow the reference stored in **this** and set it's maxHP variable

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

Stack

Name	Value
p1	
Player	
this	0x999
name	"Dark Cecil"
maxHP	10
setMaxHP	
this	0x999
maxHP	10

Heap

Player

maxHP	10
hp	
name	

0x999

in/out

Notation Note:

- I'll gray out a stack frame that is removed from the stack
- This will have the same meaning as crossing it out
- Makes the variables readable

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

Stack

Name	Value
p1	
Player: this	0x999
Player: name	"Dark Cecil"
Player: maxHP	10
setMaxHP: this	0x999
setMaxHP: maxHP	10
setHP: this	0x999
setHP: hp	10

Heap

Player

maxHP	10
hp	10
name	

0x999

in/out

- Calling setHP will set the hp variable on the stack for this object

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

Stack

Name	Value
p1	
Player: this	0x999
Player: name	"Dark Cecil"
Player: maxHP	10
setMaxHP: this	0x999
setMaxHP: maxHP	10
setHP: this	0x999
setHP: hp	10
setName: this	0x999
setName: name	"Dark Cecil"

Heap

Player

maxHP	10
hp	10
name	"Dark Cecil"

0x999

in/out

- Repeat the process for name

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

Stack

Name	Value
p1	0x999
Player	
this	0x999
name	"Dark Cecil"
maxHP	10
setMaxHP	
this	0x999
maxHP	10
setHP	
this	0x999
hp	10
setName	
this	0x999
name	"Dark Cecil"

Heap

Player

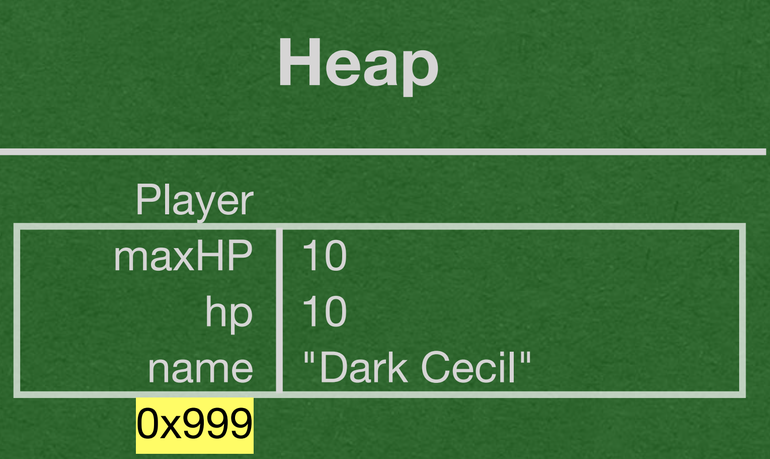
maxHP	10
hp	10
name	"Dark Cecil"

0x999

in/out

- Constructor method calls return a reference to the object that was created

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```



in/out

- What happens on the line that initializes p2?

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

- Whenever you see ***new***, a new object is created on the heap
- We have 2 objects of type Player
  - Each object has its own copies of each instance variable

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

- What happens on the line that initializes p3?

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

- If you **don't** see new, **no** object is created
- Assign p3 the same reference stored in p1
- Still only 2 objects on the heap

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

Stack

Name	Value
p1	0x999
Player: this	0x999
Player: name	"Dark Cecil"
Player: maxHP	10
setMaxHP: this	0x999
setMaxHP: maxHP	10
setHP: this	0x999
setHP: hp	10
setName: this	0x999
setName: name	"Dark Cecil"
p2	0x820
Player: this	0x820
Player: name	"Kain"
Player: maxHP	14
setMaxHP: this	0x820
setMaxHP: maxHP	14
setHP: this	0x820
setHP: hp	14
setName: this	0x820
setName: name	"Kain"
p3	0x999
setName: this	0x999
setName: name	"Paladin"

Heap

Player

maxHP	10
hp	10
name	~~"Dark Cecil"~~ "Paladin"

0x999

Player

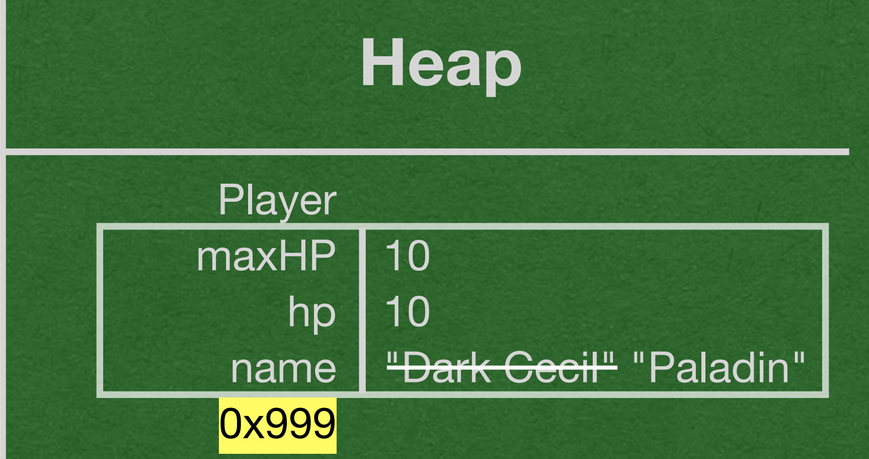
maxHP	14
hp	14
name	"Kain"

0x820

in/out

- setName is called from p1 which stores 0x999
- **this** is assigned 0x999

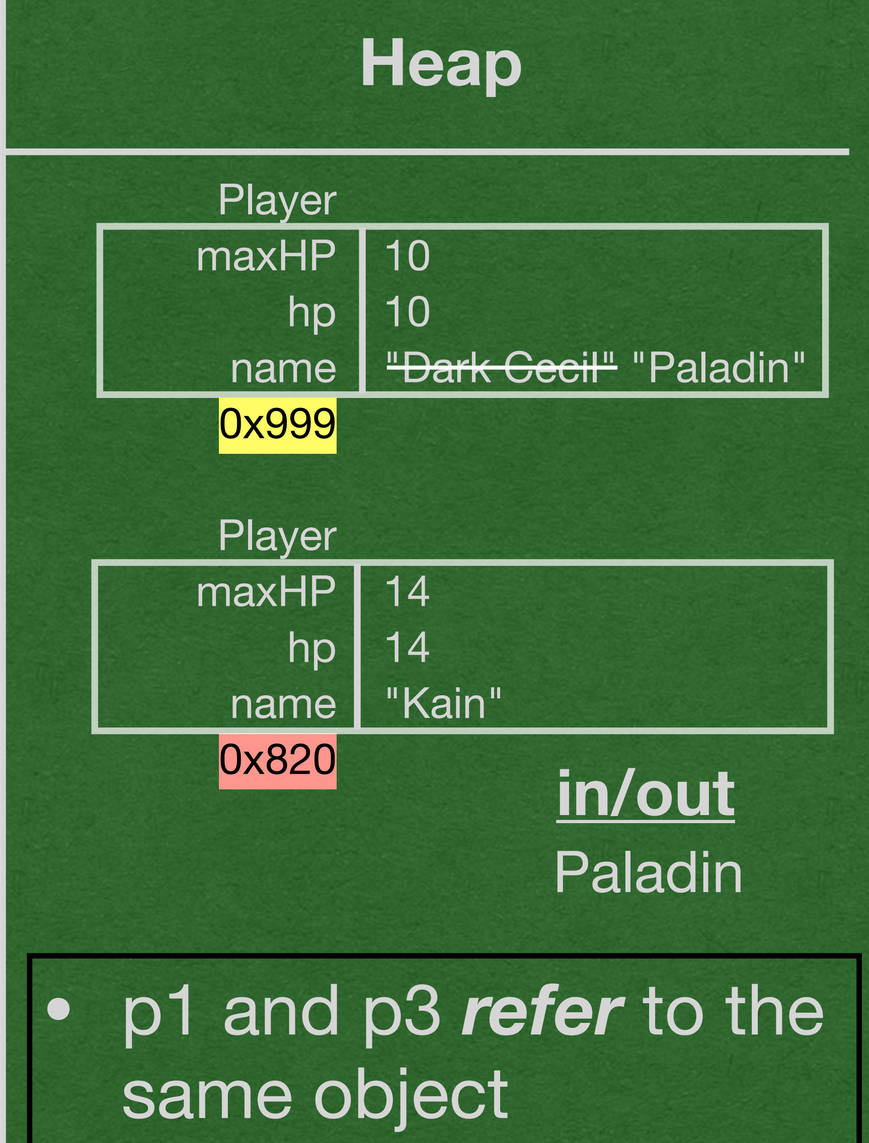
```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```



in/out

- getName is called from p3 which stores 0x999
- **this** is assigned 0x999

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```



in/out

Paladin

- p1 and p3 ***refer*** to the same object
- Any change made using one variable, affects both variables!

```java
public class Player {
    private int maxHP;
    private int hp;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public void setMaxHP(int maxHP) {
        this.maxHP = maxHP;
    }
    public void setHP(int hp) {
        if (hp <= this.maxHP) {
            this.hp = hp;
        } else {
            this.hp = this.maxHP;
        }
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p1;
        p1.setName("Paladin");
        System.out.println(p3.getName());
    }
}
```

in/out

Paladin

- End Program

Stack Memory

- Only primitive types are stored directly on the stack
  - double
  - int
  - char
  - boolean
  - String*
  - Double/Integer/Character/Boolean*

- **Everything else** is stored on the heap with only their references stored on the stack**
  - This includes **every** object created from a class that **you wrote**

***Strings and boxed types are actually more complex, but we will treat them as though they are on the stack in this course because they *behave* exactly as a value on the stack**

****Stack and heap allocations vary by compiler and JVM implementations. With modern optimizations, we can never be sure where our values will be stored We'll use this simplified view so we can move on and learn Computer Science**

toString

```java
package week3;

public class Player {

    private int maxHP;
    private int hp;
    private int attackPower = 4;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

  /** Getters and Setters removed for slide **/

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        System.out.println(p1);
    }
}
```

week3.Player@279f2327

- When you print one of your objects to the screen
  - It prints garbage
- Fully qualified name
- @
- "random" hex value
- Almost always not what you want

toString

```java
package week3;

public class Player {

    private int maxHP;
    private int hp;
    private int attackPower = 4;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public String toString() {
        String out = "health: " + this.hp + "/";
        out += this.maxHP;
        return out;
    }

  /** Getters and Setters removed for slide **/

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        System.out.println(p1);
    }
}
```

health: 10/10

- If we write a special method named "toString" that returns a String
- This method will be called when we print an object of this type

```java
public class Player {
    private int maxHP;
    private int hp;
    private int attackPower = 4;
    private String name;

    public Player(String name, int maxHP) {
        this.setMaxHP(maxHP);
        this.setHP(maxHP);
        this.setName(name);
    }

    public String toString() {
        String out = "health: " + this.hp + "/";
        out += this.maxHP;
        return out;
    }

    public void takeDamage(int damage) {
        this.hp -= damage;
    }

    public void attack(Player otherPlayer) {
        otherPlayer.takeDamage(this.attackPower);
    }

  /** Getters and Setters removed for slide **/

    public static void main(String[] args) {
        Player p1 = new Player("Dark Cecil", 10);
        Player p2 = new Player("Kain", 14);
        Player p3 = p2;
        p1.attack(p2);
        p1.attack(p2);
    }
}
```

Types

- Classes define types
- Now that we have a Player type, we can use it wherever we need a type
- Here, we use Player as the type of a method parameter

Party Example

```java
public class Character {
    private int battlesWon = 0;
    private int expPoints = 0;

    public Character() {}
    public void winBattle(int xp) {
        this.battlesWon++;
        this.expPoints += xp;
    }
}
```

```java
public class Party {
    private ArrayList<Character> members;
    private int battlesWon = 0;

    public Party() {
       this.members = new ArrayList<>();
    }
    public void addCharacter(Character member) {
        this.members.add(member);
    }
    public void winBattle(int xp) {
        this.battlesWon++;
        for (int x=0; x < this.members.size(); x++) {
            this.members.get(x).winBattle(xp);
        }
    }
    public static void main(String[] args) {
        Character hero = new Character();
        hero = new Character();
        Character fighter = new Character();
        hero.winBattle(10);
        Party party = new Party();
        party.addCharacter(hero);
        party.addCharacter(fighter);
        party.winBattle(20);
    }
}
```

Party

- Let's look at another example
- We'll highlight 2 new concepts with this code:
  - Composition
  - Garbage Collection

```java
public class Character {
    private int battlesWon = 0;
    private int expPoints = 0;

    public Character() {}
    public void winBattle(int xp) {
        this.battlesWon++;
        this.expPoints += xp;
    }
}
```

```java
public class Party {
    private ArrayList<Character> members;
    private int battlesWon = 0;

    public Party() {
       this.members = new ArrayList<>();
    }
    public void addCharacter(Character member) {
        this.members.add(member);
    }
    public void winBattle(int xp) {
        this.battlesWon++;
        for (int x=0; x < this.members.size(); x++) {
            this.members.get(x).winBattle(xp);
        }
    }
    public static void main(String[] args) {
        Character hero = new Character();
        hero = new Character();
        Character fighter = new Character();
        hero.winBattle(10);
        Party party = new Party();
        party.addCharacter(hero);
        party.addCharacter(fighter);
        party.winBattle(20);
    }
}
```

Composition

- Instance variables of objects can store references to the objects
- The Party class is composed of an ArrayList of Characters

```java
public class Character {
    private int battlesWon = 0;
    private int expPoints = 0;

    public Character() {}
    public void winBattle(int xp) {
        this.battlesWon++;
        this.expPoints += xp;
    }
}
```

```java
public class Party {
    private ArrayList<Character> members;
    private int battlesWon = 0;

    public Party() {
       this.members = new ArrayList<>();
    }
    public void addCharacter(Character member) {
        this.members.add(member);
    }
    public void winBattle(int xp) {
        this.battlesWon++;
        for (int x=0; x < this.members.size(); x++) {
            this.members.get(x).winBattle(xp);
        }
    }
    public static void main(String[] args) {
        Character hero = new Character();
        hero = new Character();
        Character fighter = new Character();
        hero.winBattle(10);
        Party party = new Party();
        party.addCharacter(hero);
        party.addCharacter(fighter);
        party.winBattle(20);
    }
}
```

Garbage Collection

- hero is assigned a reference to an instance/object of type Character
- hero is then reassigned to a new reference
- We no longer have a reference to the first Character object in memory
  - Since we cannot access this object, it will be removed from memory

Memory Diagram

```java
public class Character {
    private int battlesWon = 0;
    private int expPoints = 0;

    public Character() {}
    public void winBattle(int xp) {
        this.battlesWon++;
        this.expPoints += xp;
    }
}
```

```java
public class Party {
  private ArrayList<Character> members;
  private int battlesWon = 0;

  public Party() {
   this.members = new ArrayList<>();
  }
  public void addCharacter(Character member) {
    this.members.add(member);
  }
  public void winBattle(int xp) {
    this.battlesWon++;
    for (int x=0; x < this.members.size(); x++) {
      this.members.get(x).winBattle(xp);
    }
  }
  public static void main(String[] args) {
    Character hero = new Character();
    hero = new Character();
    Character fighter = new Character();
    hero.winBattle(10);
    Party party = new Party();
    party.addCharacter(hero);
    party.addCharacter(fighter);
    party.winBattle(20);
  }
}
```

Stack

Name	Value
hero	~~0x875~~ 0x320
Character: this	0x875
Character: this	0x320
fighter	0x555
Character: this	0x555
winBattle: this	0x320
winBattle: xp	10
party	0x746
Party: this	0x746
addCharacter: this	0x746
addCharacter: member	0x320
addCharacter: this	0x746
addCharacter: member	0x555
winBattle: this	0x746
winBattle: xp	20
winBattle: x	~~0~~ ~~1~~ 2
winBattle: this	0x320
winBattle: xp	20
winBattle: this	0x555
winBattle: xp	20

Heap

Character 0x875 (crossed out)

battlesWon	0
expPoints	0

Character 0x320

battlesWon	~~0~~ ~~1~~ 2
expPoints	~~0~~ ~~10~~ 30

Character 0x555

battlesWon	~~0~~ 1
expPoints	~~0~~ 20

Party 0x746

members	0x313
battlesWon	~~0~~ 1

0x313

0	0x320
1	0x555

in/out

- End Program