Phillip Ball

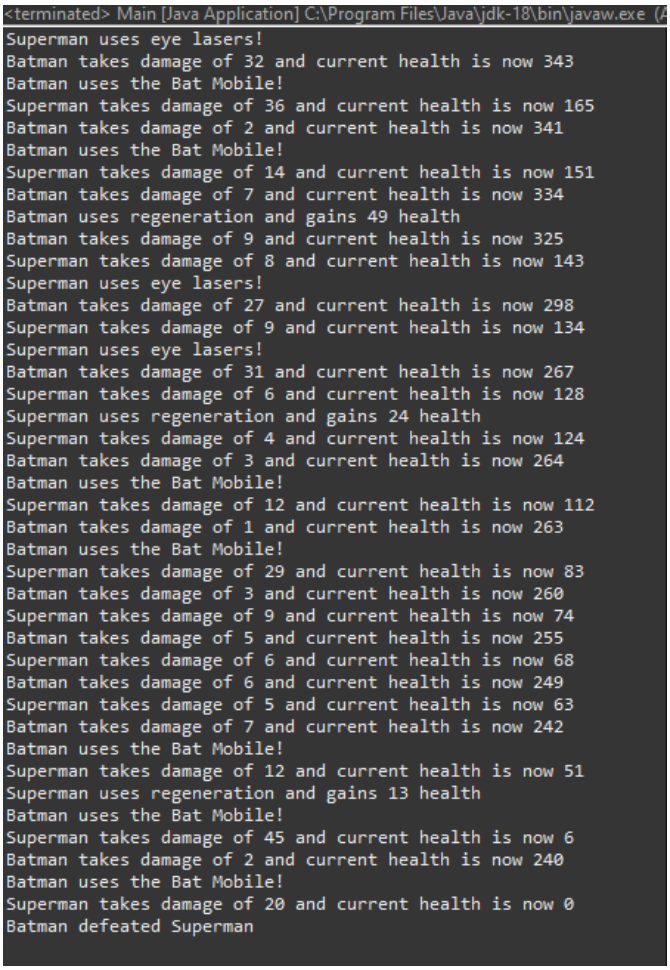
CST-239

Prof. Couch

4/21/22

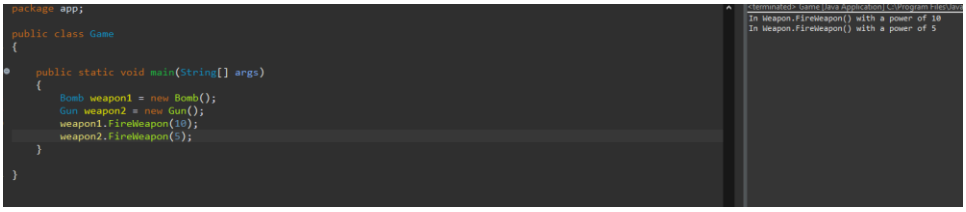
Activity 2: Inheritance

Part 1: Super Hero Battle

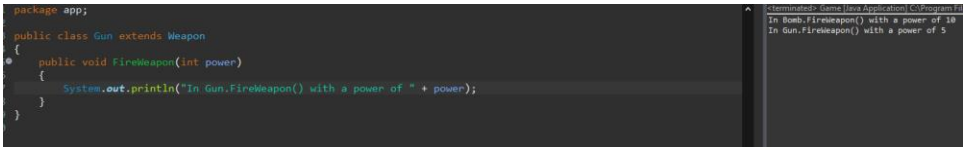


This output was displayed because in the main function I generated super heroes to fight. After creating the super heroes, I put them into a while loop that made sure they were still both alive to continue fighting, if that was the case then they would continue to attack each other. I added a special attack for each hero and a regeneration for each hero that would take place of the normal attack based on a 20% chance for each. There was a 60% chance that the attack would be normal, 20% chance that the attack would be special, and 20% chance that they would regen health.

Part 2: Weapons, Bombs, and Guns



This output was displayed because the bomb and gun class are both extensions of the weapon class. Inside the weapon class there is a function that is called FireWeapon, and using this function inside of the abstract class Weapon, I can display the function through the class extensions.



This output was displayed from overriding the main weapon function of FireWeapon. Making a FireWeapon function inside of the class extensions allows the program to take their priority over the abstract class, since it is the actual class instead. When creating the Gun weapon1 = new Gun(); we are specifically calling this class so functions inside of this class that carry the same name will be taken priority over the abstract counterpart.

This output was displayed because the program recognizes that there is not an input of an integer anymore, allowing the overloaded method to take priority instead. Then using super, we can go back to calling FIreWeapon() from the Weapon class instead inside of the actual class extension functions.

```java
package app;

public abstract class Weapon
{
    public void FireWeapon(int power)
    {
        System.out.println("In Weapon.FireWeapon() with a power of " + power);
    }
    public abstract void Activate(boolean enable);
}
```

```
In the Bomb.Activate() with an enable of true
In the Gun.Activate() with an enable of true
In Bomb.FireWeapon() with a power of 10
In Gun.FireWeapon() with a power of 5
In overloaded Bomb.FireWeapon()
In Weapon.FireWeapon() with a power of 5
In overloaded Gun.FireWeapon()
In Weapon.FireWeapon() with a power of 10
```

```java
public class Gun extends Weapon
{
    public void FireWeapon(int power)
    {
        System.out.println("In Gun.FireWeapon() with a power of " + power);
    }
    public void FireWeapon()
    {
        System.out.println("In overloaded Gun.FireWeapon()");
        super.FireWeapon(10);
    }
    public void Activate(boolean enable)
    {
        System.out.println("In the Gun.Activate() with an enable of " + enable);
    }
}
```

```java
public class Bomb extends Weapon
{
    public void FireWeapon(int power)
    {
        System.out.println("In Bomb.FireWeapon() with a power of " + power);
    }
    public void FireWeapon()
    {
        System.out.println("In overloaded Bomb.FireWeapon()");
        super.FireWeapon(5);
    }
    public void Activate(boolean enable)
    {
        System.out.println("In the Bomb.Activate() with an enable of " + enable);
    }
}
```

```java
public class Game
{

    public static void main(String[] args)
    {
        Bomb weapon1 = new Bomb();
        Gun weapon2 = new Gun();
        weapon1.Activate(true);
        weapon2.Activate(true);
        weapon1.FireWeapon(10);
        weapon2.FireWeapon(5);
        weapon1.FireWeapon();
        weapon2.FireWeapon();
    }

}
```

Problems | Debug Shell

6 errors, 8 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (6 items)				
The abstract method Activate in type Weapon can only be defined by an abstract class	Weapon.java	/topic2-2/src/app	line 9	Java Problem
The method FireWeapon(int) is undefined for the type Object	Bomb.java	/topic2-2/src/app	line 12	Java Problem
The method FireWeapon(int) is undefined for the type Object	Gun.java	/topic2-2/src/app	line 12	Java Problem
The type Bomb cannot subclass the final class Weapon	Bomb.java	/topic2-2/src/app	line 3	Java Problem
The type Gun cannot subclass the final class Weapon	Gun.java	/topic2-2/src/app	line 3	Java Problem
The type Weapon must be an abstract class to define abstract methods	Weapon.java	/topic2-2/src/app	line 3	Java Problem

These errors occurred because the other classes rely on Weapon class being an abstract class, the Activate Boolean is based from this class being abstract, so it returns an error. The gun and bomb class can no longer extend from Weapon and or use the super function.

Description	Resource	Path	Location	Type
Errors (2 items)				
Cannot override the final method from Weapon	Bomb.java	/topic2-2/src/app	line 5	Java Problem
Cannot override the final method from Weapon	Gun.java	/topic2-2/src/app	line 5	Java Problem

These errors occurred because you cannot override a final void. Final voids become static as in they cannot be changed by any extension class, every extension class would have to use the final method inside of the abstract class.

Description	Resource	Path	Location	Type
Errors (1 item)				
Abstract methods do not specify a body	Weapon.java	/topic2-2/src/app	line 5	Java Problem

This error occurred because abstract methods cannot have a body. With the Activate function, the function intakes a bool and leaves the body to be made by the extension classes.

Part 3: How to Compare Person Objects

```
<terminated> Test [Java Application] C:\Program Files\Java\jdk-18\bin\ja
These persons are not identical using ==
These persons are not identical using equals()
This copied person is not identical using equals()
app.Person@5a07e868
app.Person@76ed5528
app.Person@2c7b84de
```

The reason this output was displayed was due to the objects being created with separate instantiations. This makes the objects have separate IDs as you can see with the "@……..". Every time a new object is created it has a different hash to it. That is why none of them are equal to each other, even the identical one.

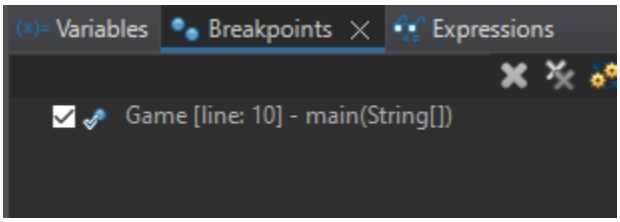
```
These persons are not identical using ==
These persons are not identical using equals()
This copied person is identical using equals()
My class is class app.Person Will Ball
My class is class app.Person Chelsea Ball
My class is class app.Person Will Ball
```

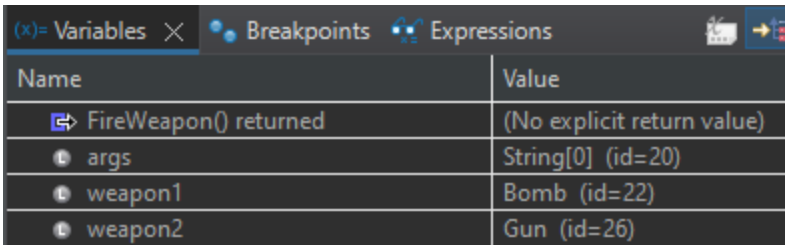
This output was displayed because instead of trying to compare the objects ID with one another, it is now comparing the objects by name and last name instead. This allows the identical copied version of Person to become identical.

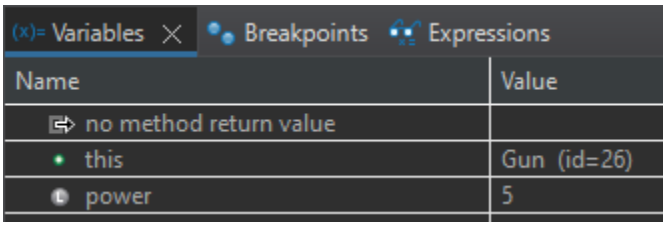
This would be very useful for the Milestone project as inside of having a bunch of items in a store, there can be duplicates that you might not want to display. Doing this would allow for the duplicates to be removed. For the project I ended up using a HashMap to loop through and find duplicates, once found I increased the stock on a product.

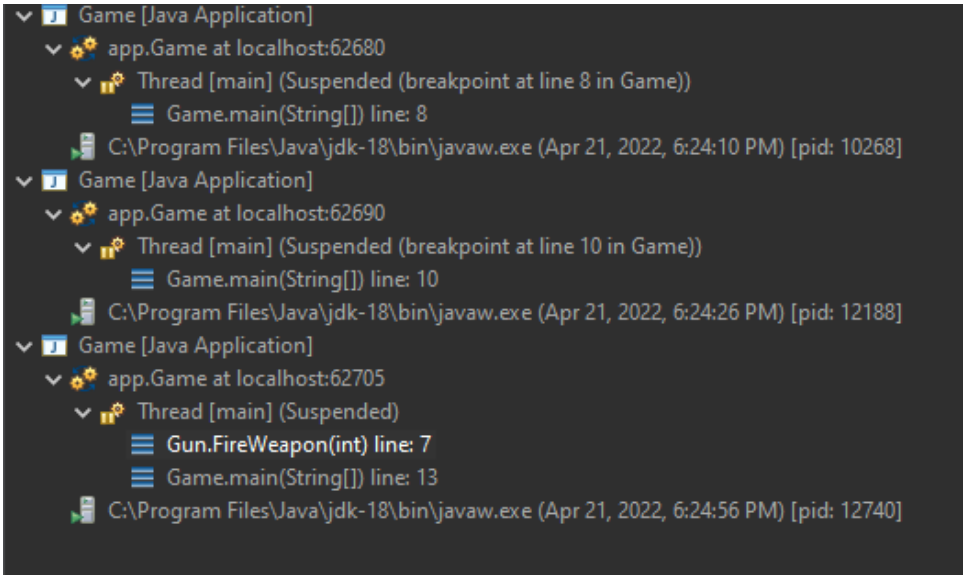
The @Override annotation lets the IDE know that the child class function is now going to overwrite its base class function. It is also useful to make sure that the function getting overridden is valid.

Part 4: Using the Debugger




Variables
Breakpoints
Expressions
Name
Value
FireWeapon() returned
(No explicit return value)
args
String[0] (id=20)
weapon1
Bomb (id=22)
weapon2
Gun (id=26)


Variables
Breakpoints
Expressions
Name
Value
no method return value
this
Gun (id=26)
power
5


Game [Java Application]
app.Game at localhost:62680
Thread [main] (Suspended (breakpoint at line 8 in Game))
Game.main(String[]) line: 8
C:\Program Files\Java\jdk-18\bin\javaw.exe (Apr 21, 2022, 6:24:10 PM) [pid: 10268]
Game [Java Application]
app.Game at localhost:62690
Thread [main] (Suspended (breakpoint at line 10 in Game))
Game.main(String[]) line: 10
C:\Program Files\Java\jdk-18\bin\javaw.exe (Apr 21, 2022, 6:24:26 PM) [pid: 12188]
Game [Java Application]
app.Game at localhost:62705
Thread [main] (Suspended)
Gun.FireWeapon(int) line: 7
Game.main(String[]) line: 13
C:\Program Files\Java\jdk-18\bin\javaw.exe (Apr 21, 2022, 6:24:56 PM) [pid: 12740]