Activity 5.3

Ramzy Izza Wardhana - 21/472698/PA/20322

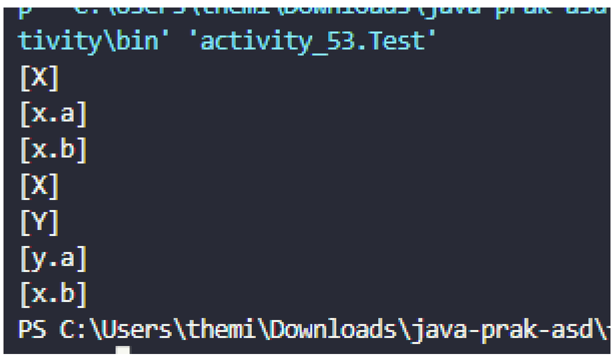
X.java

```java
package activity_53;

public class X {
    X() { System.out.println("[X]");}
    public void a() { System.out.println("[x.a]");}
    public void b() { System.out.println("[x.b]");}
    X(int i) { System.out.println("[X(int i)]");}

}

class Y extends X {
    Y() { System.out.println("[Y]"); }
    public void a() { System.out.println("[y.a]");}
    Y(int i) { System.out.println("[Y(int i)]");}
}

class Z extends Y {}
```

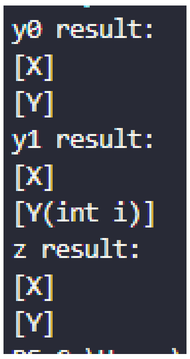
Test.java

```java
public class Test {
    public static void main(String[] args) {
        X x = new X();
        x.a();
        x.b();

        Y y = new Y();
        y.a();
        y.b();
    }
}
```

Output



```
tivity\bin' 'activity_53.Test'
[X]
[x.a]
[x.b]
[X]
[Y]
[y.a]
[x.b]
PS C:\Users\themi\Downloads\java-prak-asd\
```



```
y0 result:
[X]
[Y]
y1 result:
[X]
[Y(int i)]
z result:
[X]
[Y]
```

Reasoning:

1. The program starts in a procedural way in which we first create a new instance from X class called x in the code `X x = new X();`
2. Next, when creating this new object, java will automatically create the “default” constructor with no parameter, which in this case, we declared to output `X() { System.out.println("[X]");}`
3. This will result in output [X] in the very beginning since every time we compile and run the program, the first initialization that java do is to run the constructor first, which in this particular case, contains `{System.out.println("[X]");}.` And therefore, resulting in the output [X].
4. Observe that, after creating the new instance of x, we call the method x.a(), and x.b(), which is a void method that will display `("[x.a]") & ("[x.b]")` respectively.
5. Furthermore, we implement an inheritance approach on class Y, which extended from class X. The general idea behind this is to inherit all the methods from x to y.
6. As a result, by constructing a new instance of y, and by invoking the methods a (method overriding) and b which are derived from x, the program will execute the output.
7. It is worth noting that the method y.a() has been overridden by the `public void a() { System.out.println("[y.a]"); }` , resulting in the final output of `"[y.a]"` instead of `"[x.a]”` . Meanwhile, this does not apply to the y.b(), and thus the program will still obtain the original method derived from x.
8. Previously, we mentioned that the constructor always comes first, and this also applies to y instances. Since the y instance is inherited from x, the “default” constructor of y will also contain “`System.out.println("[X]");`” followed with `System.out.println("[y]");.` This underlying logic would give the elaboration on why the output `"[X]"` happened twice before other y methods run.
9. Similarly to the next test, every inherited instance or method that we invoked, which is the result of the derivation from x, would yield the same concept, in which java will virtually add the x default constructor to the child constructor, hence for every method, will result `[X]` in the very beginning.