Manavendra Sen | 199302058 | IT 3A

OOP LAB 3

Static blocks

Static blocks are also called Static initialization blocks. A static initialization block is a normal block of code enclosed in braces, { }, and preceded by the static keyword.

```
class StaticExample {

 static {
   System.out.println("This is first static block");
 }

 public StaticExample() {
   System.out.println("This is constructor");
 }

 public static String staticString = "Static Variable";
 static {
   System.out.println("This is second static block and " + staticString);
 }

 public static void main(String[] args) {
   StaticExample statEx = new StaticExample();
   StaticExample.staticMethod2();
 }

 static {
   staticMethod();
   System.out.println("This is third static block");
 }

 public static void staticMethod() {
   System.out.println("This is static method");
 }
```

```
public static void staticMethod2() {
  System.out.println("This is static method2");
 }
}
```

What will happen when you execute the above code? Run the code and observe the output.

First all static blocks are positioned in the code and they are executed when the class is loaded into JVM. StaticMethod2() static method is executed after the class is instantiated because it is being called after the instantiation of the class.

```
mavn:~/ $ cd Java/Practicals/DAY2                    [13:43:05]
mavn:DAY2/ $ javac LAB\ 3.java                       [13:43:17]
mavn:DAY2/ $ java StaticExample                      [13:43:23]
This is first static block
This is second static block and Static Variable
This is static method
This is third static block
This is constructor
This is static method2
mavn:DAY2/ $                                         [13:43:26]
```

EXAMPLE 5
Consider the class Circle. It has methods to compute the radius and circumference of a circle.

```
class Circle {
 // static final means this is a class attribute and its value can't be changed
 static final double PI = 3.14;
 private double radius;

 Circle(double radius) {
   this.radius = radius;
 }

 // accessor method
 public double getRadius() {
   return radius;
 }

 // mutator method
 public void setRadius(double radius) {
   this.radius = radius;
 }

 // method to find the area
 public static double area(Circle c1) {
   // Static reference to PI
   return (PI * c1.getRadius() * c1.getRadius());
 }

 // static method's are not instance specific
 public static void getCircumference(double radius) {
   // here radius variable is a local variable
   System.out.println("Circumference = " + 2 * PI * radius);
 }
}// End of circle
```

```java
class TestCircle {
 public static void main(String args[]) {
   Circle c1 = new Circle(2.3);
   Circle.area(c1);

   // accessing static method with class name
   Circle.getCircumference(2.3);
   Circle c2 = new Circle(3.45);
   Circle.area(c2);

   // accessing static method with references is discouraged
   // c1.getCircumference(3.45); // this will work with warnings, static
methods should be referenced statically
   Circle.getCircumference(3.45);
 } // end of main
} // end of class
```

```
mavn:DAY2/ $ javac CircleLab3.java                    [13:46:23]
mavn:DAY2/ $ java TestCircle                          [13:46:46]
Circumference = 14.443999999999999
Circumference = 21.666
mavn:DAY2/ $                                          [13:47:07]
```