25/07/22

JAVA ASSIGNMENT

By Devayush Bajaj

1)WAP of static keyword with all possible ways. (variable,method,class, block)

```
//Rule no 1 = always use static method into the static method
//static is called as initializer block, and it is also known as load
initializer block
//within static block we can use local and static variable

public class Static_uses {
    String name = "Devayush";            //Non-Static local variable
   static int aadhar_card = 12345;     //Static local variable

   //NON-Static method
   public void Non_Static_accept_data(){
       System.out.println(name +" "+ aadhar_card);
   }

   //static method

   public static void static_show_data(){
       //System.out.println(name  + aadhar_card);        // Name cannot be
fetched because name variable is static in nature
       System.out.println("Name will not display " +
"Aadhar_card"+aadhar_card);
   }

   //Static block
   //First priority
   static{                //only need main method and don't need to mention
under main, Static gets more priority than main but it required main method in
program
       String non_static_variable = "Under static block";
       System.out.println(non_static_variable);
   }
   //Static class
   static class Dev_static_class {
       static void Display(){
           System.out.println("This method is under static class");
       }
   }

   public static void main(String[] args) {

       Static_uses static_uses = new Static_uses();
```

```
        static_uses.Non_Static_accept_data();        //calling non static method

        static_show_data();      //calling static method under non static class

        Dev_static_class.Display(); //calling static method under static class


    }
}
```

```
//Rule no 1 = always use static method into the static method
 //static is called as initializer block, and it is also known as load initializer block
//within static block we can use local and static variable

public class Static_uses {
     1 usage
     String name = "Devayush";            //Non-Static local variable
    2 usages
    static int aadhar_card = 12345;     //Static local variable

    //NON-Static method
    1 usage
    public void Non_Static_accept_data(){
        System.out.println(name +" "+ aadhar_card);
    }

    //static method

    1 usage
    public static void static_show_data(){
        //System.out.println(name  + aadhar_card);        // Name cannot be fetched because name variable is static
        System.out.println("Name will not display " + "Aadhar_card"+aadhar_card);
    }
```

```
    //Static block
    //First priority
    static{                                  //only need main method and don't need to mention under main, Stati
        String non_static_variable = "Under static block";
        System.out.println(non_static_variable);
    }
    //Static class
    1 usage
    static class Dev_static_class {
        1 usage
        static void Display(){
            System.out.println("This method is under static class");
        }
    }

    public static void main(String[] args) {

        Static_uses static_uses = new Static_uses();
        static_uses.Non_Static_accept_data();       //calling non static method

        static_show_data();             //calling static method under non static class

        Dev_static_class.Display();   //calling static method under static class
```

```
    }
}
```

2)Explore =>storage of static methods and static variables in Java

- Static methods (in fact all methods) as well as static variables are stored in the PermGen section of the heap, since they are part of the reflection data (class related data, not instance related).

Note that only the variables and their technical values (primitives or references) are stored in PermGen space.

- If your static variable is a reference to an object, that object itself is stored in the normal sections of the heap (young/old generation or survivor space). Those objects (unless they are internal objects like classes etc.) are not stored in PermGen space.
- Prior to Java 8:
- The static variables were stored in the permgen space(also called the method area).
- PermGen Space is also known as Method Area

PermGen Space used to store 3 things
Class level data (meta-data)
interned strings
static variables

- The static variables are stored in the Heap itself.From Java 8 onwards the PermGen Space have been removed and new space named as MetaSpace is introduced which is not the part of Heap any more unlike the previous Permgen Space. Meta-Space is present on the native memory (memory provided by the OS to a particular Application for its own usage)and now it only stores the class meta data.

3)Prove practically=>
3.1) Can we Overload static methods in Java

YES, we can

```java
public class CAN_WE_OVERLOAD_STATIC {
    static void Details(String name){
        System.out.println("Name: "+ name);
    }
    static void Details (String name, int id){
        System.out.println("Name " + name + "\n"+ "ID " + id );
    }
    CAN_WE_OVERLOAD_STATIC can_we_overload_static = new CAN_WE_OVERLOAD_STATIC();

    public static void main(String[] args) {
        Details( name: "Devayush");
        Details( name: "Devayush",  id: 15);
    }
}

/*
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Ed
Name: Devayush
Name Devayush
```

OUTPUT:

```
"C:\Program Files\Java\jdk1.8.0_
Name: Devayush
Name Devayush
ID 15
```

3.2)Can we Override static methods in Java

we cannot override static methods because method overriding is based on dynamic binding at runtime and the static methods are bonded using static binding at compile time.

```java
public class CAN_WE_OVERRIDE_STATIC {
    static void Details(String name){
        System.out.println("Name is "+ name);
    }
    static void Details(String address){
        System.out.println("Address is "+ address);
    }

    public static void main(String[] args) {
        Details( name: "Devayush");

    }
}
```

The calling of method depends upon the type of object that calls the static method. It means:

- If we call a static method by using the parent class object, the original static method will be called from the parent class.

- If we call a static method by using the child class object, the static method of the child class will be called.

3.3)Why main() method is declared as static?

Java main() method is always static, so that compiler can call it without the creation of an object or before the creation of an object of the class.

- In any Java program, the main() method is the starting point from where compiler starts program execution. So, the compiler needs to call the main() method.
- If the main() is allowed to be non-static, then while calling the main() method JVM has to instantiate its class.
- While instantiating it has to call the constructor of that class, There will be ambiguity if the constructor of that class takes an argument.
- Static method of a class can be called by using the class name only without creating an object of a class.
- The main() method in Java must be declared public, static and void. If any of these are missing, the Java program will compile but a runtime error will be thrown.

4)Is there any error in the below code snippet? If yes, identify the error and give the reason behind it.

```
public class Demo
{
void m1(Demo demo) {
System.out.println("Instance method");
}
static void m1(Demo d) {
System.out.println("Static method");
}
}
```

```java
class Demo {

        1 usage
        void m1(Demo demo) {
            System.out.println("Instance method");
        }
        1 usage
        static void m2(Demo d) {
            System.out.println("Static method");
        }
    }

class Find_The_Error{
    public static void main(String[] args) {
        Demo demo = new Demo();
        demo.m1(demo);
        Demo.m2(demo);
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
Instance method
Static method

Process finished with exit code 0
```

ERROR: - Error was in the method created. The methods contained the same name with the same number of parameters. Which leads to violating ambiguity.

5)Will the following code snippet compile fine? If yes, what will be output

```
public class Myclass{
private static int x=10;
static {
X++;
static {
++x;
}
 {
X--
}
public static void main(String[] args) {
Myclass obj = new Myclays();
Myclass obj2 = new Myclass();
```

```java
public class Will_This_compile{
    3 usages
    private static int x=10;
    static{
        x++;
    }
    {
        x--;
    }
    public static void main(String [] args){
        Will_This_compile obj1 = new Will_This_compile();
        Will_This_compile obj2 = new Will_This_compile();
        Will_This_compile obj3 = new Will_This_compile();
        System.out.println(x);
    }

}

// OUTPUT - 8
```

OUTPUT:

```
Run: Will_This_compile
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
8

Process finished with exit code 0
```