Static Keyboard Assignment Questions

Assignment Questions:

1. Why do we need static keyword in Java Explain with an example?

Ans → The static keyword in Java is used to indicate that a member (field or method) of a class belongs to the class itself, rather than to an instance of the class. In other words, a static member is shared across all instances of the class.

a. To create class-level variables: When a variable is declared as static, it is created once for the entire class, rather than once for each instance of the class. This can be useful for creating constants or for sharing data across all instances of the class.

```
public class MyClass {
    public static final double PI = 3.14159;
    public static int count = 0;
}
```

In this example, the PI constant is a class-level variable that is shared across all instances of the MyClass class. The count variable is also a class-level variable, but it can be modified by any instance of the class.

b. To create class-level methods: A static method is a method that belongs to the class itself, rather than to any instance of the class. This can be useful for creating utility methods that don't need to access instance-level data.

```
public class MathUtils {
    public static int add(int a, int b) {
        return a + b;
    }
}
```

In this example, the add method is a class-level method that can be called directly on the MathUtils class, without creating an instance of the class.

2. What is class loading and how does the java program actually executes?

Ans → Class loading is the process by which Java runtime environment loads Java class files into memory. Java programs are compiled into Java bytecode, which is then interpreted and executed by the Java Virtual Machine (JVM). When a Java program is run, the JVM uses class loading to load the necessary class files into memory.

The class loading process has three steps:

a. Loading: The first step is to locate the bytecode for the class and load it into memory. The bytecode may be located on the local file system or on a remote server. The class

loader searches the classpath, which is a list of directories and JAR files that contain Java class files.

b. Linking: The second step is to link the bytecode to the rest of the Java runtime environment. This includes verifying the bytecode for correctness, preparing the memory for the class variables and methods, and resolving any dependencies on other classes.
c. Initialization: The final step is to initialize the class by executing the code in the class's static initializer block, which is a special block of code that is executed only once, when the class is first loaded.

Once the class has been loaded, linked, and initialized, it is ready to be used by the Java program.

3. Can we mark a local variable as static?

Ans → Yes, we can mark a local variable as static in Java. However, doing so is not allowed in Java syntax, as static keyword is used to declare class-level variables, methods, and nested classes. Static keyword can't be used with local variables.

In Java, local variables are defined within the method or block, and their scope is limited to that method or block only. Local variables are created when a method is called and destroyed when the method returns. Therefore, marking a local variable as static doesn't make any sense, as the purpose of the static keyword is to create a variable or method that is shared by all instances of the class.

If you need to declare a variable that retains its value between method calls, you can declare it as an instance variable, or as a class variable (static variable) if it needs to be shared by all instances of the class.

4. Why is the static block executed before the main method in java?

Ans → In Java, a static block is executed before the main method because the static block is part of the class loading process, while the main method is executed when the program is run.
When a Java class is loaded, the class loader initializes the static variables and executes any static blocks in the class. This happens only once, when the class is loaded, and it ensures that the static variables are initialized before any instance of the class is created or any static method is called.
On the other hand, the main method is the entry point of the Java program and is executed when the program is run. The main method is called after the class is loaded and any static blocks have been executed.
Therefore, the order of execution is:

a. The class is loaded
b. Any static blocks in the class are executed
c. The main method is executed

This order of execution ensures that any static variables are initialized before the main method is executed, which is necessary for the correct functioning of the program.

5. Why is a static method is also called a class method?

Ans → A static method is also called a class method because it belongs to the class rather than an instance of the class.

In Java, a static method is declared using the static keyword and can be called without creating an object of the class. This means that the method belongs to the class and not to any particular instance of the class.
Since a static method is associated with the class itself, it can be accessed using the class name, followed by a dot (.) and then the name of the method. This is different from instance methods, which can only be called on an instance of the class.
For example, consider a class called Math which has a static method called max that returns the maximum of two numbers. This method can be called using the class name Math.max(a, b), without creating an instance of the Math class.
Therefore, a static method is also called a class method because it belongs to the class and can be accessed using the class name.

6. What is the use of static blocks in Java?

Ans →In Java, static blocks are used to initialize static variables or perform any other one-time initialization tasks that are required for the class to function properly.
A static block is a block of code that is executed only once when the class is loaded into memory. The purpose of the static block is to perform any initialization tasks that cannot be done using a static initializer or a constructor.

7. Difference between Static and Instance variables.

Ans →In Java, static variables and instance variables are two types of class variables. They differ in their scope, storage, and lifetime.

a. Scope: Static variables have a class-level scope, meaning they are shared by all instances of the class. Instance variables have object-level scope, meaning they are unique to each instance of the class.
b. Storage: Static variables are stored in the method area of memory, while instance variables are stored in the heap memory.
c. Lifetime: Static variables are created when the class is loaded into memory and destroyed when the program exits. Instance variables are created when an object is created and destroyed when the object is garbage collected.
d. Initialization: Static variables can be initialized using a static initializer or by assigning a value to them at the time of declaration. Instance variables are typically initialized using a constructor or by assigning a value to them at the time of declaration.
e. Access: Static variables can be accessed using the class name, followed by a dot (.) and then the name of the variable. Instance variables are accessed using the object name, followed by a dot (.) and then the name of the variable.

8. Difference between Static and Static members.

Ans → In Java, static and static members are related concepts, but they refer to different things.

Static refers to a keyword that can be used to modify variables, methods, and blocks, while static members refer to variables and methods that are declared with the static keyword.

Here are some differences between static and static members:

a. Usage: Static is used to modify variables, methods, and blocks, while static members refer specifically to static variables and methods.
b. Scope: Static has a scope that depends on where it is used, while static members have a class-level scope. This means that static members are shared by all instances of the class.
c. Memory allocation: Static variables and methods are allocated memory in the method area of memory, while instance variables and methods are allocated memory on the heap.
d. Access: Static members can be accessed using the class name, followed by a dot (.) and then the name of the member. Instance members can be accessed using the object name, followed by a dot (.) and then the name of the member.
e. Initialization: Static members can be initialized using a static initializer or by assigning a value to them at the time of declaration. Instance members are typically initialized using a constructor or by assigning a value to them at the time of declaration.