OOPS Fundamentals Assignment Solution

Question 1: How to Create an Object in Java?

Answer 1: The Object is a basic building of an OOP's language. In, Java we cannot execute any program without creating an object. There is various way to create an object in Java.

- Java provides five ways to create an object.
  - Using new Keyword
  - Using clone() method
  - Using newInstance() method of the Class class
  - Using newInstance() method of the Constructor class
  - Using Deserialization
- Using new Keyword
  - Using the new keyword is the most popular way to create an object or instance of the class. When we create an instance of the class by using the new keyword, it allocates memory (heap) for the newly created object and also returns the reference of that object to that memory. The new keyword is also used to create an array. The syntax for creating an object is:
  - ClassName object = new ClassName();

Question 2: What is the use of new keyword in Java?

Answer 2: Using the new keyword is the most popular way to create an object or instance of the class. When we create an instance of the class by using the new keyword, it allocates memory (heap) for the newly created object and also returns the reference of that object to that memory. The new keyword is also used to create an array. The syntax for creating an object is:

- ClassName object = new ClassName();

Question 3: What are the different types of variables in Java?

Answer 3: Variable: A variable is the name of a reserved area allocated in memory. In other words, it is a name of the memory location. It is a combination of "vary + able" which means its value can be changed.

- Types of Variables:
  - There are three types of variables in Java:
    - local variable
    - instance variable
    - static variable

- Local Variable: A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists. A local variable cannot be defined with "static" keyword.
- Instance Variable: A variable declared inside the class but outside the body of the method, is called an instance variable. It is not declared as static. It is called an instance variable because its value is instance-specific and is not shared among instances.
- Static variable: A variable that is declared as static is called a static variable. It cannot be local. You can create a single copy of the static variable and share it among all the instances of the class. Memory allocation for static variables happens only once when the class is loaded in the memory.

Question 4: What is the difference between Instance variable and Local variable?

Answer 4:

Instance Variable	Local Variable
They are defined in class but outside the body of methods.	They are defined as a type of variable declared within programming blocks or subroutines.
These variables are created when an object is instantiated and are accessible to all constructors, methods, or blocks in class.	These variables are created when a block, method or constructor is started and the variable will be destroyed once it exits the block, method, or constructor.
These variables are destroyed when the object is destroyed.	These variables are destroyed when the constructor or method is exited.
It can be accessed throughout the class.	Its access is limited to the method in which it is declared.
They are used to reserving memory for data that the class needs and that too for the lifetime of the object.	They are used to decreasing dependencies between components I.e., the complexity of code is decreased.
These variables are given a default value if it is not assigned by code.	These variables do not always have some value, so there must be a value assigned by code.
It is not compulsory to initialize instance variables before use.	It is important to initialize local variables before use.

Question 5: In which area memory is allocated for instance variable and local variable?

Answer 5: Instance Variable:

- Instance variables will be stored on the heap as the part of the object.

- Instance variables should be declared within the class directly but outside of any method or block or
- constructor.
- Instance variables can be accessed directly from the Instance area. But cannot be accessed directly from a
- static area.
- But by using object reference we can access instance variables from static area.
- For every object a separate copy of instance variables will be created.
- Objects and corresponding instance variables will be stored in the Heap area.

Local Variable:

- Local variables will be stored inside the stack.
- The local variables will be created as part of the block execution in which it is declared and destroyed once that block execution completes. Hence the scope of the local variables is exactly the same as the scope of the block in which declared.
- The local variables will be stored on the stack.
- For the local variables JVM won't provide any default values, we should perform initialization explicitly before using that variable.
- For every method the JVM will create a runtime stack, all method calls performed by that Thread and corresponding local variables will be stored in that stack. Every entry in stack is called Stack Frame or Action record.

Question 6: What is Method Overloading?

Answer 6: In Java, Method Overloading allows different methods to have the same name, but different signatures where the signature can differ by the number of input parameters or type of input parameters, or a mixture of both.

- Method overloading in Java is also known as Compile-time Polymorphism, Static Polymorphism, or Early binding. In Method overloading compared to the parent argument, the child argument will get the highest priority.
- Different Ways of Method Overloading in Java
    - Changing the Number of Parameters.
    - Changing Data Types of the Arguments.
    - Changing the Order of the Parameters of Methods
- Changing the Number of Parameters: Method overloading can be achieved by changing the number of parameters while passing to different methods.
- Changing Data Types of the Arguments: In many cases, methods can be considered Overloaded if they have the same name but have different parameter types, methods are considered to be overloaded.

- Changing the Order of the Parameters of Methods: Method overloading can also be implemented by rearranging the parameters of two or more overloaded methods. For example, if the parameters of method 1 are (String name, int roll_no) and the other method is (int roll_no, String name) but both have the same name, then these 2 methods are considered to be overloaded with different sequences of parameters.