**Assignment – 2**

1. Polymorphism is the concept that objects and classes will take on the role best suited for them at the time they are needed. Or in simple terms, the same interface with different functionalities that activate as and when they are needed.
2. Inheritance refers to the transfer of attributes, values and structures from one set of objects and classes to the other.
3. Abstraction is the concept of showing only what is necessary for a user to make a program work for them, and not revealing the internal details. The colloquial term for this is Black Box Programming.
4. Encapsulation is the concept of ensuring that objects, attributes, properties and structures of a program are not affected by unauthorized factors, and which only interact with the outside world as and when appropriate.
5. Aggregation is a specialized form of association between two or more objects in which the objects have their own life-cycle but there exists an ownership as well. Aggregation is a typical whole/part kind of relationship, but it may or may not denote physical containment -- the objects may or may or be a part of the whole. In aggregation the objects have their own life-cycle, but they do have ownership as well.
6. Composition is a specialized form of aggregation in which if the parent object is destroyed, the child objects would cease to exist.
7. Association is a semantically weak relationship (a semantic dependency) between otherwise unrelated objects. An association is a "using" relationship between two or more objects in which the objects have their own life time and there is no owner.
8. The **default** class modifier is a relationship restriction which ensures that the members of the same class and package (as well as that of the super-class) can access it and its components but restricts access beyond that unless explicitly defined. In other words, **Protected** mode with a slight allowance for sub-class access.
9. **Public, Private, Protected, Default, Final.**
10. The **final** modifier on a class ensures that it may not be inherited and may not be extended.
11. The **final** modifier on a method ensures that it cannot be overridden, and its elements are uninheritable.
12. The **java.lang** package is always imported by default.
13. The Packages/Classes specified may be imported as many times as desired, but the JVM will load them up only once.
14. No. Importing a package allows only all its classes and interfaces available to user - not its sub-packages. To avail the class and interfaces of the sub-packages they have to be explicitly imported.
15. A Package is a mechanism to encapsulate a group of classes, sub packages and interfaces. It is always the first thing loaded up by the JVM, and it serves as the defining encapsulation for the program.
16. Packages are used for –
    1. Preventing naming conflicts.
    2. Making searching and locating classes, interfaces, enumerations and annotations much easier.
    3. Controlling access to Package members for security and stability.
    4. Advanced form of Encapsulation.
17. Yes, imports are checked for their semantic validity at compile time. Any code contained above the line of imports will not compile.
18. The differences are as follows –
    1. Public – All access permitted
    2. Protected – Access permitted within the class, package and subclasses of the protected class.
    3. Default – Access permitted within the class, package and the subclasses within the same package.
    4. Private – Access permitted within the class only.
19. The access modifier of an overriding or hiding method must provide at least as much access as the overridden or hidden method, or a compile-time error occurs. In detail –
    1. If the overridden or hidden method is public, then the overriding or hiding method must be public; otherwise, a compile-time error occurs.
    2. If the overridden or hidden method is protected, then the overriding or hiding method must be protected or public; otherwise, a compile-time error occurs.
    3. If the overridden or hidden method has default (package) access, then the overriding or hiding method must not be private; otherwise, a compile-time error occurs.
20. **Public, Private, Protected.** No modifier implies **Default.**
21. **See Answer 19.**
22. It means that without reflection, the constructor won’t be accessible outside of the class. Other classes cannot call it. Only members of the constructor's class will be able to create its object.
23. Yes, but only it’s subclass will be able to instantiate. Non-subclasses have no access.
24. No. Since private methods can’t be overridden, they can’t be made abstract either.
25. **strictfp** is a keyword in the Java programming language that restricts floating-point calculations to ensure portability.
26. The **instanceof** operator is used to test whether the object is an instance of the specified type (class or subclass or interface). It is also known as a type comparison operator because it compares the instance with its type. It returns either true or false. If applied to any variable that has null value, it returns false.