1-Define Class.?

A **class** is nothing but a blueprint or a template for creating different objects which defines its properties and behaviors. **Java class** objects exhibit the properties and behaviors **defined** by its **class**. A**class** can contain fields and methods to describe the behavior of an object.

2-Define Object..

Object – Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An **object** is an instance of a class. Class – A class can be **defined** as a template/blueprint that describes the behavior/state that the **object** of its type support.

3-Instance.?

The new operator instantiates a class by allocating memory for a new object and returning a reference to that memory. Note: The phrase "instantiating a class" means the same thing as "creating an object." When you create an object, you are creating an "**instance**" of a class, therefore "instantiating" a class.

4-Encapsulation.?

Encapsulation is one of the four **fundamental** OOP concepts. The **other** three are inheritance, polymorphism, and abstraction. Encapsulation in Java is a mechanism of **wrapping** the data (variables) and code acting on the data (methods) together as a single unit.

5-Information Hiding.?

Information hiding is a powerful programming technique because it reduces complexity. One of the chief mechanisms for **hiding information** is encapsulation -- combining elements to create a larger entity. The programmer can then focus on the new object without worrying about the hidden details.

6-Constructor.?

Constructor java tutorial: Java constructors are the methods which are used to initialize objects. Constructor method has the **same** name as that of class, they are called or invoked when an object of class is created and can't be called explicitly.

7-Constructor Overloading.?

Constructor Overloading in Java. Constructor overloading is a technique in **Java** in which a **class** can have any number of **constructors** that differ in parameter lists.The compiler

differentiates these**constructors** by taking into account the number of parameters in the list and their type.

8-Inheritance.?

Inheritance is one of the feature of Object-Oriented Programming (OOPs).**Inheritance** allows a class to use the properties and methods of another class. ... This is because **Java** does not support multiple **inheritance**. The superclass and subclass have "is-a" relationship between them.

9.Plateform Independance.?

platform independence means that the same program works on any **platform** (operating system) without needing any modification. In the case of **Java** the application runs in a **Java** Virtual Machine which itself isn't**platform independent**..

10-Byte Code.?

Programming **code** that, once compiled, is run through a virtual machine instead of the computer s processor. ... **Bytecode** is the compiled format for **Java** programs. Once a **Java** program has been converted to **bytecode**, it can be transferred across a network and executed by Java Virtual Machine (JVM).

11-Function Overloading.?

Method Overloading is a feature that allows a class to have two or more methods having same name, if their argument lists are different. In the last tutorial we discussed constructor **overloading** that allows a class to have more than one constructors having different argument lists.

12-Function Over-Riding.?

Overriding in Java. ... When a **method** in a subclass has the same name, same parameters or signature and same return type(or sub-type) as a **method** in its super-class, then the**method** in the subclass is said to **override** the **method** in the super-class.

13-Access Specifiers.?

Java Access Specifiers (also known as Visibility **Specifiers**) regulate **access** to classes, fields and methods in **Java**.These **Specifiers**determine whether a field or method in a class, can be used or invoked by another method in another class or sub-class. **Access Specifiers** can be used to restrict **access.** **TYPE: Private, Public, Protected.**

14-Final Data Members.?

Final variables are often declared with the static keyword in **Java** and are treated as constants. For example: ... Note: A class declared as **final** cannot be extended or inherited (i.e, there cannot be a subclass of the super class). It is also good to note that **methods** declared as **final** cannot be overridden by subclasses.

15-Final Classes.?

Writing **Final Classes** and **Methods**. You can declare some or all of a **class's methods final**. You use the **final** keyword in a **method** declaration to indicate that the **method** cannot be overridden by subclasses. The Object **class** does this—a number of its **methods** are **final** .

16- Concrete Class.?

A **concrete class** is a **class** that has an implementation for all of its **methods** that were inherited from abstract or implemented via interfaces. It also does not **define**any abstract **methods** of its own. ... Therefore it can be inferred that any **class** that is not an abstract **class** or interface is a **concrete class**.

17-Package.?

A package is a namespace that organizes a set of related classes and interfaces. Conceptually you can think of packages as being **similar** to different folders on your computer.