OOPS

(1) what is inheritence in Java?

Ans:- The technique of creating a new class by existing class functionality is called inheritence in Java. In other words, Inheritence is a process where a child class ~~acqr~~acquires all the properties and behaviours of the parent Class.

(2) superclass and subclass?

Ans:- A class from where a subclass inherits ~~from~~ features is called superclass. It is also called base class or parent class.

A class that inherits all the members (field. method and nested classes) from another class is called a subclass. It is also called as child class, derived class or extends class

(3) How is inheritence implemented / achieved in Java?

Ans:- Inheritence can be implemented or achieved by using two keywords.

extends:- extends is a keyword that is used for developing inheritance between two classes and two interfaces.

Implements:- Implements keyword is used for developing the ~~interface~~ inheritence between a class and interface.

(4) What is polymorphism ?

Ans:- polymorphism in OOP is the ability of an entity to teke several forms. In the other words it refers to the ability of an object (or a reference to an object) to teke different forms of objects. It allows to a common data gathering message to be sent to each class. polymorphism encourages what is called extendibility which means an Objects or a class can have its uses extened.

(5) Difference between method overloading and overriding :

overriding	overloading
• Implements runtime polymorphism.	• Implement's compile time polymorphism
• The method call is determine at runtime base ont the object type.	• Method call is determined at compile time.
• occurs between superclass and subclass	• occurs between the methods the same class
• Have the same signature (name and method arguemts).	• Have the same name, but the parameters are different
• on error, the effect will be visible at runtime	• on error, it can be caught at compile time

If we try to inherit final class, then the compiler throws an error at complication time. we can't create a class as immutable without final class.

```
final class parent class {
      void showData()
        {
        syso("This is a method of final parent class");
        }
}
   class child class extends parent class
  {
       void ~~main()~~ showdata
     {
       syso("This is a method of child class");
  }
  }
class mainclass
    {
       ps vm (String[] args){
    {
       parent class obj = new child class();
         obj. showDate();
    }
  }.
```

(10) Difference between Runtime and compile time polymorphism Explain with an example.

Compile Time polymorphism	Runtime polymorphism
• compile time polymorphism is less flexible as all things executed at compile time.	• Runtime polymorphism is more flexible as all thing execute at runtime.
• In compile time polymorphism, the call is resolved by compiler.	• In Runtime polymorphism the call is not resolved by compiler.
• Inheritence is not involved	• Inheritence is involved.
• It is also known as static binding, Early binding and overloading as well	• It is also known as dynamic binding, Late binding and overriding as well.
• It provides fast execution because the method that needs to be executed is known early at the compile time.	• It provides slow execution as compared to early binding because the method that needs to be executed is known at runtime.
• method overloading is compile time polymorphism where more than one method shares the same name with different parameters or signature and different return type.	• method overriding is the runtime polymorphism having the same method with same parameter or signature but associate with compared, different classes

$\frac{1}{29}$ $\frac{1}{(4/3)^2}$ $\left(\frac{1}{\frac{4}{3}}\right)^2$ $\left(\frac{16}{9}\right)$

(10)

(9) Difference Between anbstraction and encapsulation

Abstraction	Encapsulation
• Anstraction is a feature of oops that hides the unnecessary detail but show the essential information	• Encapsulation is also a feature of oops. It hides a code and data to a single entity or unit so that the data can be protected from the outside world
• It solves an issue at the design level.	• Encapsulation solves an issue at implementation level.
• It focuses on the external lookout.	• It focus on internal working.
• It can be implemented using abstract classes and interface.	• It can be implemend by using the access modifier (private, public, protected)
• It is the process of gaining Information	• It is the process of containing the information
• In abstraction we use abstract classes and interface. to hide the code complexities.	• we use the getter and setter methods to hide the data
• The objects are encapsulated that helps to perform abstraction.	• The object need not to abstract that result in encapsulation

(6) what is abstraction explain with an example?

Ans:- Abstraction is nothing but the quality of dealing with ideas rathater than events. It basically deals with hidding the internal class and showing the essential thing to the user

```
Abstract class sports { // abstract class sports
Abstract void jump(); // abstract method
}
```

(7) what is differente between an abstract method and final method in java? Explain with an example

Ans:- The abstract method is incomplete while the final method is regarded as complete. The only way to use an abstract method is by overriding it, but you can not override a final method in java.

(8) what is final Class in java?

Ans:- A class declared with the final keyword is known as the final class. A final class can't be inherited by subclasses. By using final class, we restrict the inheritence of the class. we can create a class as a final class only if it is complete in nature. which means it must not be an abstract class. In Java, all the wrapper classes are final classes like String Integer etc.