1)what is Abstraction

Abstraction is the process of hiding the internal implementation details and showing only the essential features .

Ex:-

abstract class Animal {

abstract void makeSound(); // abstract method

}

class Dog extends Animal {

void makeSound() {

System.out.println("Bark");

}

}

2. Difference Between Interface and Abstract Class

Abstarct class :-

Abstract class can have both bstact methods and concrete methods and abstract class can have constructors and any access modifier

Abstract class not supported multiple inheritance can have variables (non final)

Interface :-

Interface can have only abstract methods (java 7)

Default and static methods allowed (java 8) and interface cannot have constructors All members are public by default interface can supported multiple inheritance .

Variable only public static final fiels.

3. Explain Polymorphism with Example?

Polymorphism means many forms . It allows the same method or object to behave differently in different contexts .

There are two types of polymorphism :

Compile –time (Method Overloading )

Runtime (Method Overriding )

Example:-

class Animal {

void makeSound() {

System.out.println("Animal sound");

}

}

class Cat extends Animal {

void makeSound() {

System.out.println("Meow");

}

}

4. What is Method Overriding?

Method Overriding means redefining a method In a child class that already exits in the parent class with same name and same parameters datatype .

class Parent {

void greet() {

System.out.println("Hello from Parent");

}

}

class Child extends Parent {

@Override

void greet() {

System.out.println("Hello from Child");

}

}

5. Explain “IS-A” vs “HAS-A” Relationships

Is –A Relationship means Inheritance relationship

Dog IS –A Animal

HAS –A Relationship means composition (user )relationship Library HAS-A Book

IS –A uses extends HAS-A uses object referenccces.

6. Why Use Inheritance?

Reusability of code

Easy maintenance

Supports IS-A relationship

Promotes code hierarchy and better structure

7. What is Dynamic Binding?

Dynamic Binding (Late Bindinng )means method calls are resolved at runtime rather than compile –time .

Used with method overriding .

Animal a = new Cat(); // Animal reference, Cat object

a.makeSound(); // Calls Cat's method (runtime decision)

8. What is Constructor Chaining?

Calling one constructor from another constructor in same class or superclass using this() or super () constructor.

class Person {

Person() {

this("Anonymous"); // Calls parameterized constructor

}

Person(String name) {

System.out.println("Name: " + name);

}

}

9. How to Implement Encapsulation?

Incapsulation means hide data keeps variables as aprivate and use getters and setters

**class Student {**

**private int age;**

**public void setAge(int age) {**

**this.age = age;**

**}**

**public int getAge() {**

**return age;**

**}**

**}**

10. Explain super Keyword?

Reference to the parent class used to call parent constructor :super() and call parent method using super.method(); and access parent variables .

class Parent {

int a = 10;

}

class Child extends Parent {

void display() {

System.out.println(super.a); // Access parent variable

}

}