Q1 answer any two of following:

a. What is **this** in java?

There can be a lot of usage of java this keyword. In java, this is a reference variable that refers to the current object.

b. What is **polymorphism** in java?

Polymorphism in java is a concept by which we can perform a single action by different ways

c. Why use Java **encapsulation**?
Encapsulation in java is a process of wrapping code and data together into a single unit, For example capsule i.e. mixed of several medicines.

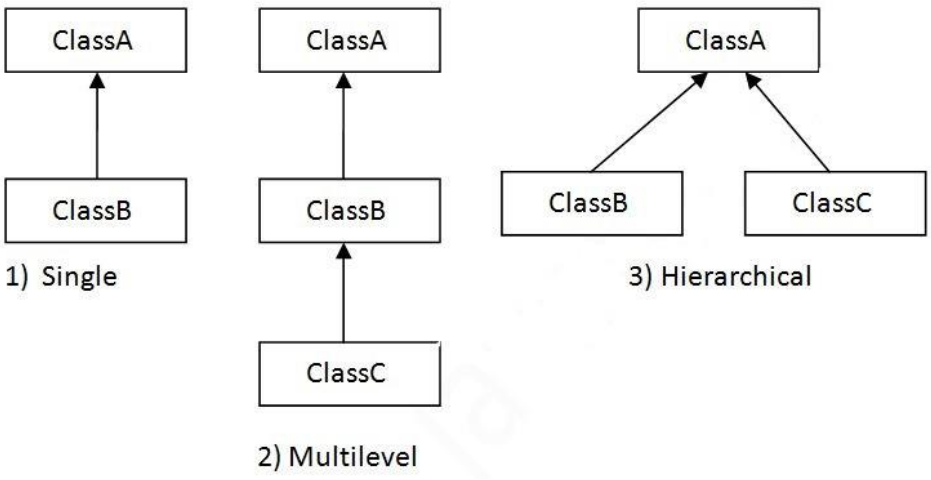
Q2 what is an **inheritance** and which types are supported in java? Explain that with a source code example.

Solution:-

Inheritance in java is a mechanism in which one object acquires all the properties and behaviors of parent object.

The idea behind inheritance in java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of parent class, and you can add new methods and fields also.

There can be three types of inheritance in java: single, multilevel and hierarchical.



Single Inheritance Example

```
1. class Animal{
2. void eat(){System.out.println("eating...");}
3. }
4. class Dog extends Animal{
5. void bark(){System.out.println("barking...");}
6. }
7. class TestInheritance{
8. public static void main(String args[]){
9. Dog d=new Dog();
10. d.bark();
11. d.eat();
12. }}
```

Multilevel Inheritance Example

```
1. class Animal{
2. void eat(){System.out.println("eating...");}
3. }
4. class Dog extends Animal{
5. void bark(){System.out.println("barking...");}
6. }
7. class BabyDog extends Dog{
8. void weep(){System.out.println("weeping...");}
9. }
10. class TestInheritance2{
11. public static void main(String args[]){
12. BabyDog d=new BabyDog();
13. d.weep();
14. d.bark();
15. d.eat();
16. }}
```

Hierarchical Inheritance Example

```
1. class Animal{
2. void eat(){System.out.println("eating...");}
3. }
4. class Dog extends Animal{
```

```
5.  void bark(){System.out.println("barking...");}
6.  }
7.  class Cat extends Animal{
8.  void meow(){System.out.println("meowing...");}
9.  }
10. class TestInheritance3{
11. public static void main(String args[]){
12. Cat c=new Cat();
13. c.meow();
14. c.eat();
15. }}
```

Q3 write a program that add names in **arrayList** and then display it on the console?

Solution:-

```
1.  import java.util.*;
2.  class TestCollection1{
3.   public static void main(String args[]){
4.    ArrayList<String> list=new ArrayList<String>();
5.    list.add("radhwan");
6.    list.add("ali");
7.    list.add("noor");
8.    list.add("sura");
9.    Iterator itr=list.iterator();
10.   while(itr.hasNext()){
11.    System.out.println(itr.next());
12.   }
13.  }
14. }
```

Q4 Using **abstraction concept on superclass:** write a program for employee class contains the variable (ID, name, mobile-no and salary) do the following operations:

1. Initial values for 4 Employee.
2. Display the information of Employee who has name = Ahmad.

Solution:-

```
package uot;
abstract public class Studentt {
   int ID;
   String name;
   long MobileNo;
   float Salary;

abstract void insert(int ID, String name, long MobileNo, float Salary);

abstract void display();
}

class student extends Studentt{
   int ID;
   String name;
   long MobileNo;
   float Salary;

void insert(int ID, String name, long MobileNo, float Salary){
   this.ID=ID;
   this.name=name;
   this.MobileNo=MobileNo;
   this.Salary=Salary;
   }

void display(){
   if (name=="Ahmad") {
   System.out.println(ID+" "+name+" "+MobileNo+" "+Salary);
   }
}
   public static void main(String[] args) {
      student s=new student();
      student s1=new student();
```

```
        student s2=new student();
        student s3=new student();

   s.insert(10,"Ali",07701111110,80.000f);
   s1.insert(11,"Ahmad",0770111111,50.000f);
   s2.insert(13,"sara",07701111112,90.000f);
   s3.insert(16,"lee",07701111113,100.000f);
        System.out.println("==========the Employee who has name = Ahmed is ========");
    s.display();
    s1.display();
    s2.display();
    s3.display();
   }
}
```

Q5 Using polymorphism and inheritance: write a program to display the names of shapes (rectangle, square and circle)?

Solution:-

```
package uot;
class Shape{
void area(){
   System.out.println("shape");
}}

class rectangle extends Shape{
void area(){
   System.out.println("rectangle");
}
}
class square extends Shape{
void area(){
   System.out.println("square");
}
}
class circule extends Shape{
void area(){
   System.out.println("circule");
}
}
```

```
class test1{
public static void main(String args[]){
Shape s;
s= new rectangle();
s.area();
s= new square();
s.area();
s= new circule();
s.area();
}
}
```

Q6 Using **constructor and this keyword:** write a program for student class (roll-no, name, subject and mark) do the following operations:

1. Initial values for 4 students.
2. Display the information of student who has roll-no = 10.
3. Make carve 2 marks for student who has mark < 5.

Solution:-

```
package uot;
class student {
   int rollno;
   String name;
   String subject;
   int mark;

 student(int rollno, String name, String subject, int mark){
   this.rollno=rollno;
   this.name=name;
   this.subject=subject;
   this.mark=mark;
   }

void display(){
   if (rollno==10) {
   System.out.println(rollno+" "+name+" "+subject+" "+mark);
   }
  }

void carve(){
```

```
    if (mark<5){
        mark=mark+2;
        System.out.println("have carve 2 marks for student "+" "+rollno+" "+name+" "+subject+"
"+mark);
    }
    else    {
        System.out.println("no mark for student "+"  "+rollno+" "+name+" "+subject+" "+mark);
    }
}
    public static void main(String[] args) {
        student s=new student(10,"ali","oop",6);
        student s1=new student(12,"rad","oop",3);
        student s2=new student(13,"sara","oop",2);
        student s3=new student(16,"lee","oop",7);

        System.out.println("==========the student who have roll-no = 10 is ============");
    s.display();
    s1.display();
    s2.display();
    s3.display();
        System.out.println("==========the marks <5 after carve==================");
    s.carve();
    s1.carve();
    s2.carve();
    s3.carve();
    }
}
```