WEEK-9

Question: Create a class Vehicle, write a method cost() in this class. Create two classes Bus and Train which have their own display() methods and inherit from Vehicle class. Create objects of Bus and Train class and call cost() and display() methods.

Code:

```
// Vehicle, Bus and Train example
class Vehicle {
void cost() {
        System.out.println("Cost varies based on vehicle type.");
    }
}
class Bus extends Vehicle {
void display() {
        System.out.println("This is a Bus.");
    }
}
class Train extends Vehicle {
void display() {
        System.out.println("This is a Train.");
    }
}
public class MainVehicle {     public
static void main(String[] args) {
        Bus b = new Bus();
Train t = new Train();
        b.cost(); b.display();
        t.cost(); t.display();
    }
}
```

Output:

Output:

Cost varies based on vehicle type.

This is a Bus.

Cost varies based on vehicle type.

This is a Train.

Question: Create class University which has data members- name and ranking. Create class Faculty that extends University class has data member- name and method- Details(). Create a new class Department which is derived from Faculty and has data member- name, chairman and method- Details() and Display() where Display() method calls Details() methods of both Faculty and Department class in its body. Create an object of Department class to Display() method and University ranking.

Code:

```
class University {
   String name = "ABC University";
   int ranking = 1;
}
class Faculty extends University {
String name = "Computer Science";
   void Details() {
      System.out.println("Faculty: " + name);
   }
}
class Department extends Faculty {
   String name = "IT";
String chairman = "Dr. Ali";
void Details() {
      System.out.println("Department: " + name + ", Chairman: " + chairman);
   }
   void Display() {
```

```
        super.Details();       /
this.Details();
        System.out.println("University Ranking: " + ranking);
    }
}
 public class MainUniversity {     public
static void main(String[] args) {
Department d = new Department();
        d.Display();
    }
}
```

Output:

Output:

Faculty: Computer Science

Department: IT, Chairman: Dr. Ali

University Ranking: 1

Question: Create class Account (Data members- Id, Account_holder_name, Address; Methods deposit(), withdraw()). Create two static methods in Account calculateSimpleInterest() and calculateCompoundInterest() and implement them.

Code: class

```
Account {
    int Id;
    String Account_holder_name;
    String Address;
    // constructors and instance methods     void deposit(double amt) {
System.out.println("Deposited: " + amt); }     void withdraw(double amt) {
System.out.println("Withdrawn: " + amt); }     static double
calculateSimpleInterest(double p, double r, double t) {         return
(p*r*t)/100.0;
    }
```

```
   static double calculateCompoundInterest(double p, double r, double t) {
return p * Math.pow(1 + r/100.0, t) - p;
   }
}
public class MainAccount {     public
static void main(String[] args) {
      System.out.println("Simple Interest = " + Account.calculateSimpleInterest(1000,5,2));
      System.out.println("Compound Interest = " + Account.calculateCompoundInterest(1000,5,2));
   }
}
```

Output:

Output:

Simple Interest = 100.0

Compound Interest = 102.5

Question: Create class Account (Data members- Id, Account_holder_name, Address; Methods deposit(), withdraw()). Declare deposit() and withdraw() as abstract methods. Declare Account class as abstract. (Create constructor in Account as well)

Code:

```
abstract class
AccountAbs {
   int Id;
   String Account_holder_name;
   String Address;
   AccountAbs(int id, String name, String addr) {
      Id = id; Account_holder_name = name; Address = addr;
   }
   abstract void deposit(double amt);
abstract void withdraw(double amt);
}
class Saving extends AccountAbs {
```

```
    double balance = 0;
   Saving(int id, String name, String addr) { super(id,name,addr); }
void deposit(double amt) { balance += amt; }     void
withdraw(double amt) { balance -= amt; }     void display() {
System.out.println("Saving Balance = " + balance); }
 }
 public class MainAbstractAccount {
   public static void main(String[] args) {
Saving s = new Saving(1, "Ali", "City");
       s.deposit(2000);
       s.withdraw(500);
       s.display();
   }
 }
```

Output:

Output:

Saving Balance = 1500.0

Question: Create two children of Account- Saving (Data Members- Min_balance; Methods display(), deposit(), withdraw()) and Current (Data Members- Max_withdrawl_limit; Methods-display(),deposit(), withdraw()) . Create constructors for both classes. Implementation of deposit() and withdraw() should be specific to Saving and Current class. Create objects of Saving and Current class and display them.

Code: abstract class

```
AccountBase {
   int Id;
   String Account_holder_name;
   String Address;
   AccountBase(int id, String name, String addr) { Id=id; Account_holder_name=name; Address=addr;
}
```

```
   abstract void deposit(double amt);
abstract void withdraw(double amt);
 }
 class SavingAcc extends AccountBase {
double balance=0;    double
Min_balance=500;
   SavingAcc(int id,String name,String addr){ super(id,name,addr);}    void
deposit(double amt){ balance+=amt; }    void withdraw(double amt){
if(balance-amt>=Min_balance) balance-=amt; }    void display(){
System.out.println("Saving Balance = " + balance); }
 }
 class CurrentAcc extends AccountBase {
double balance=0;    double
Max_withdrawl_limit=10000;
   CurrentAcc(int id,String name,String addr){ super(id,name,addr);}    void
deposit(double amt){ balance+=amt; }    void withdraw(double amt){
if(amt<=Max_withdrawl_limit) balance-=amt; }    void display(){
System.out.println("Current Balance = " + balance); }
 }
 public class MainAccounts {    public static void
main(String[] args) {        SavingAcc s = new
SavingAcc(1,"Ali","City");
      s.deposit(2000); s.withdraw(1000); s.display();
CurrentAcc c = new CurrentAcc(2,"Sara","Town");
      c.deposit(5000); c.withdraw(2000); c.display();
   }
 }
```

Output:

Output:

Saving Balance = 1000.0

Current Balance = 3000.

Question: Create a class Shape with a method area(). Create two derived classes Rectangle and Circle that extend Shape. Each class should override the area() method to calculate the area of the respective shape. Create objects of Rectangle and Circle and call their area() methods.

Code:

```
class Shape {
void area() {
      System.out.println("Area method in Shape");
   }
}
class Rectangle extends Shape {
int length, breadth;
   Rectangle(int l, int b){ length=l; breadth=b; }
   @Override    void area(){ System.out.println("Area of Rectangle = " + (length
* breadth)); }
}
class Circle extends Shape {
   int radius;
   Circle(int r){ radius=r; }
   @Override    void area(){ System.out.println("Area of Circle = " + (3.14 *
radius * radius)); }
}
public class MainShape {    public
static void main(String[] args) {
      Rectangle r = new Rectangle(5,4);
Circle c = new Circle(3);
      r.area();
      c.area();
   }
}
```

Output:

Output:

Area of Rectangle = 20

Area of Circle = 28.26

Question: Create a class Employee with data members: name, salary, and a method showDetails(). Create a class Manager that extends Employee with an additional data member department. Override the showDetails() method in Manager to display all details, including department. Create an object of Manager and call showDetails().

Code:

```
class Employee {
String name;     double
salary;
   Employee(String n, double s){ name=n; salary=s; }     void showDetails(){
System.out.println("Name: " + name + ", Salary: " + salary); }
}
class Manager extends Employee {
   String department;
   Manager(String n, double s, String d){ super(n,s); department=d; }
   @Override
   void showDetails(){ System.out.println("Name: " + name + ", Salary: " + salary + ", Department: " +
department); }
}
public class MainEmployee {
   public static void main(String[] args) {
      Manager m = new Manager("Aisha", 75000, "HR");
      m.showDetails();
   }
}
```

Output:

Output:

Name: Aisha, Salary: 75000.0, Department: HR

Question: Create an abstract class Appliance with data members brand, power and abstract methods turnOn() and turnOff(). Create two derived classes WashingMachine and Refrigerator that provide their own implementations of turnOn() and turnOff(). Create objects of WashingMachine and Refrigerator and call their methods.

Code: abstract class

```
Appliance {     String
brand;     int power;
   Appliance(String b, int p){ brand=b; power=p; }
abstract void turnOn();     abstract void turnOff();
}
class WashingMachine extends Appliance {
WashingMachine(String b,int p){ super(b,p); }
   void turnOn(){ System.out.println(brand + " WashingMachine turned ON. Power: " + power + "W"); }     void turnOff(){ System.out.println(brand + " WashingMachine turned OFF."); }
}
class Refrigerator extends Appliance {     Refrigerator(String b,int p){ super(b,p); }     void turnOn(){ System.out.println(brand + " Refrigerator turned ON. Power: " + power + "W"); }     void turnOff(){ System.out.println(brand + " Refrigerator turned OFF."); }
}
public class MainAppliance {     public static void main(String[] args) {
      WashingMachine wm = new WashingMachine("LG",1000);
Refrigerator rf = new Refrigerator("Samsung",150);
wm.turnOn(); wm.turnOff();        rf.turnOn(); rf.turnOff();
   }
}
```

Output:

Output:

LG WashingMachine turned ON. Power: 1000W

LG WashingMachine turned OFF.

Samsung Refrigerator turned ON. Power: 150W

Samsung Refrigerator turned OFF.

Question: Create a class MathOperations with two static methods: findGCD(int a, int b) to calculate the greatest common divisor and findLCM(int a, int b) to calculate the least common multiple. Call these methods without creating an object of MathOperations.

Code:

```
class MathOperations {
static int findGCD(int a, int b){
if(b==0) return a;        return
findGCD(b, a%b);
   }
   static int findLCM(int a, int b){
return (a / findGCD(a,b)) * b;
   }
   public static void main(String[] args){
      System.out.println("GCD of 12 and 18 = " + findGCD(12,18));
      System.out.println("LCM of 12 and 18 = " + findLCM(12,18));
   }
}
```

Output:

Output:

GCD of 12 and 18 = 6

LCM of 12 and 18 = 36

Question: Create a class Student with data members rollNo, name, marks. Add a static variable schoolName. Create static method changeSchoolName() to update schoolName. Demonstrate how the static variable is shared among all objects.

Code:

```
class Student {
int rollNo;
String name;    int
marks;
   static String schoolName = "ABC School";
   Student(int r,String n,int m){ rollNo=r; name=n; marks=m; }
static void changeSchoolName(String s){ schoolName = s; }
   void display(){ System.out.println("Roll: "+rollNo+", Name: "+name+", Marks: "+marks+", School: "+schoolName); }    public static void
main(String[] args){
      Student s1 = new Student(1,"Zara",85);
Student s2 = new Student(2,"Omar",90);
      s1.display();
s2.display();
      Student.changeSchoolName("XYZ High");
      s1.display();
s2.display();
   }
}
```

Output:

Output:

Roll: 1, Name: Zara, Marks: 85, School: ABC School

Roll: 2, Name: Omar, Marks: 90, School: ABC School

Roll: 1, Name: Zara, Marks: 85, School: XYZ High

Roll: 2, Name: Omar, Marks: 90, School: XYZ Hig