WEEK-11

Question: Create an interface Account having methods deposit(), withdraw() and a static method aboutBank(). Create two classes Saving and Current which implement the Account interface. Call the methods of Saving and Current classes in main().

Code:

```
interface Account {
void deposit(double amt);
void withdraw(double amt);
static void aboutBank(){
      System.out.println("This is ABC Bank.");
   }
}
class Saving implements Account {    double balance = 0;    public
void deposit(double amt){ balance += amt; }    public void
withdraw(double amt){ balance -= amt; }    void display(){
System.out.println("Saving Balance = " + balance); }
}
class Current implements Account {    double balance = 0;    public
void deposit(double amt){ balance += amt; }    public void
withdraw(double amt){ balance -= amt; }    void display(){
System.out.println("Current Balance = " + balance); }
}
public class MainAccount1 {    public
static void main(String[] args){
      Saving s = new Saving();
Current c = new Current();
      s.deposit(2000); s.withdraw(500); s.display();
      c.deposit(5000); c.withdraw(1000); c.display();
      Account.aboutBank();
   }
}
```

Output:

Output:

Saving Balance = 1500.0

Current Balance = 4000.0

This is ABC Bank.

Question: Extend previous question: Add a default method takeLoan() in Account interface. It should be implemented only by Saving class. Call all methods in main().

Code:

```
interface AccountNew {    void deposit(double amt);    void
withdraw(double amt);    static void aboutBank(){
System.out.println("This is ABC Bank."); }    default void takeLoan(){
System.out.println("Loan facility available."); }
}
class SavingNew implements AccountNew {
  double balance=0;    public void deposit(double amt){ balance+=amt; }
public void withdraw(double amt){ balance-=amt; }    public void takeLoan(){
System.out.println("Saving Account Loan Approved."); }    void display(){
System.out.println("Saving Balance = "+balance); }
}
class CurrentNew implements AccountNew {
  double balance=0;    public void deposit(double amt){
balance+=amt; }    public void withdraw(double amt){ balance-
=amt; }    void display(){ System.out.println("Current Balance =
"+balance); } }

public class MainAccount2 {    public
static void main(String[] args){
SavingNew s=new SavingNew();
CurrentNew c=new CurrentNew();
```

```
    s.deposit(3000); s.withdraw(500); s.display(); s.takeLoan();
    c.deposit(4000); c.withdraw(1000); c.display();
    AccountNew.aboutBank();
  }
}
```

Output:

Output:

Saving Balance = 2500.0

Saving Account Loan Approved.

Current Balance = 3000.0

This is ABC Bank.

Question: Create interfaces Bike and Scooty, both having methods offer() and default method details(). Create class BuySomething implementing both. Resolve ambiguity by overriding details(), calling both interface versions. Call methods of BuySomething in main().

Code:

```
interface Bike {    void offer();    default void details(){
System.out.println("Bike details..."); }
}
interface Scooty {
  void offer();   default void details(){
System.out.println("Scooty details..."); }
}
class BuySomething implements Bike, Scooty {   public void offer(){
System.out.println("Festival offer on two-wheelers!"); }   public void
details(){
    Bike.super.details();
    Scooty.super.details();
  }
```

```
  public static void main(String[] args){
BuySomething b=new BuySomething();
      b.offer();
      b.details();
  }
}
```

Output:

Output:

Festival offer on two-wheelers!

Bike details...

Scooty details...

Question: Create two interfaces Printer and Scanner, both having methods connect() and default method details(). Create MultiFunctionMachine class implementing both. Override details() to resolve ambiguity and call both versions. Call all methods in main().

Code:

```
interface Printer {      void connect();      default void details(){
System.out.println("Printer details..."); }
}
interface Scanner {      void connect();      default void details(){
System.out.println("Scanner details..."); }
}
class MultiFunctionMachine implements Printer, Scanner {
  public void connect(){ System.out.println("Machine connected."); }
  public void details(){
     Printer.super.details();
     Scanner.super.details();
  }
  public static void main(String[] args){
     MultiFunctionMachine m=new MultiFunctionMachine();
     m.connect();
```

```
        m.details();
    }
}
```

Output:

Output:

Machine connected.

Printer details...

Scanner details...

Question: Create interface Device with powerOn(). Create interface SmartDevice extending Device and adding connectWiFi(). Create class SmartPhone implementing SmartDevice. Demonstrate calling both methods in main().

Code:

```
interface Device {
void powerOn();
}
interface SmartDevice extends Device {
void connectWiFi();
}
class SmartPhone implements SmartDevice {     public void
powerOn(){ System.out.println("Phone Powered ON"); }     public void
connectWiFi(){ System.out.println("WiFi Connected"); }     public
static void main(String[] args){        SmartPhone sp=new
SmartPhone();        sp.powerOn();        sp.connectWiFi();
    }
}
```

Output:

Output:

Phone Powered ON

WiFi Connected