\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WEEK-5\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

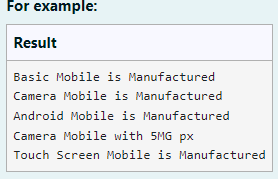
ROLL NO:230701233

1. Create a class Mobile with constructor and a method basicMobile().

Create a subclass CameraMobile which extends Mobile class , with constructor and a method newFeature().

Create a subclass AndroidMobile which extends CameraMobile, with constructor and a method androidMobile().

display the details of the Android Mobile class by creating the instance.



CODE:

class mob{ mob(){

System.out.println("Basic Mobile is Manufactured");

}

void basmob(){

System.out.println("Basic Mobile is Manufactured");

}

}

class cam extends mob{ cam(){

super();

System.out.println("Camera Mobile is Manufactured");

}

void newm(){

System.out.println("Camera Mobile with 5MG px");

}

}

class and extends cam{ and(){

super();

System.out.println("Android Mobile is Manufactured");

}

void andmob(){

System.out.println("Touch Screen Mobile is Manufactured");

}

}

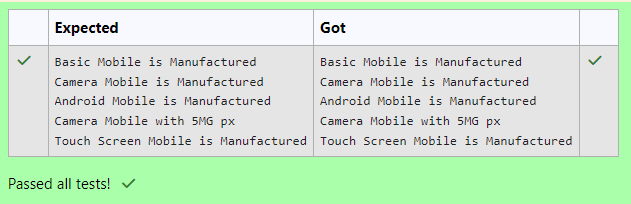
public class Main{

public static void main(String[]args){ and andmob=new and();

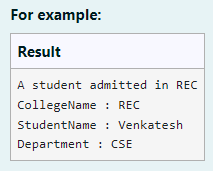
andmob.newm(); andmob.andmob();

}

} OUTPUT:



1. create a class called College with attribute String name, constructor to initialize the name attribute , a method called Admitted(). Create a subclass called CSE that extends Student class, with department attribute , Course() method to sub class. Print the

details of the Student.  CODE:

class College

{

public String collegeName;

public College(String collegeName) {

// initialize the instance variables this.collegeName=collegeName;

}

public void admitted() {

System.out.println("A student admitted in "+collegeName);

}

}

class Student extends College{

String studentName;

String department;

public Student(String collegeName, String studentName,String department) {

// initialize the instance variables super(collegeName);

this.studentName=studentName; this.department=department;

}

public String toString(){

// return the details of the student

return "CollegeName : "+collegeName+"\n"+"StudentName : "+studentName+"\n"+"Department : "+department;

}

}

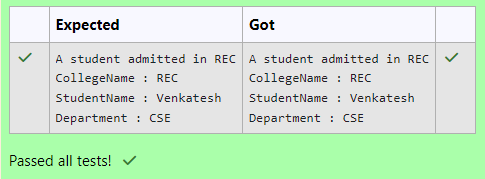
public class Main {

public static void main (String[] args) {

Student s1 = new Student("REC","Venkatesh","CSE"); s1.admitted(); // invoke the admitted() method System.out.println(s1.toString());

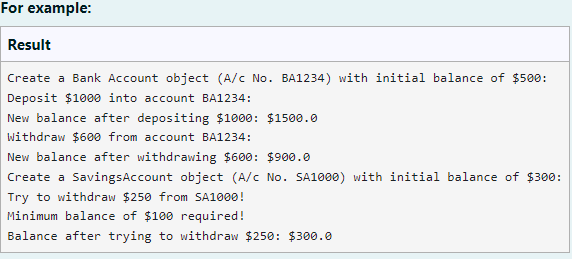
}

} OUTPUT:



1. Create a class known as "BankAccount" with methods called deposit() and withdraw().

Create a subclass called SavingsAccount that overrides the withdraw() method to prevent withdrawals if the account balance falls below one hundred.



CODE:

class BankAccount {

// Private field to store the account number private String accountNumber;

// Private field to store the balance private double balance;

// Constructor to initialize account number and balance public BankAccount(String accountNumber,double balance){

this.accountNumber=accountNumber; this.balance=balance;

}

// Method to deposit an amount into the account public void deposit(double amount) {

// Increase the balance by the deposit amount balance+=amount;

}

// Method to withdraw an amount from the account public void withdraw(double amount) {

// Check if the balance is sufficient for the withdrawal if (balance >= amount) {

// Decrease the balance by the withdrawal amount balance -= amount;

} else {

// Print a message if the balance is insufficient System.out.println("Insufficient balance");

}

}

// Method to get the current balance public double getBalance() {

// Return the current balance

return balance;

}

public String getAccountNumber(){ return accountNumber;

}

}

class SavingsAccount extends BankAccount {

// Constructor to initialize account number and balance

public SavingsAccount(String accountNumber, double balance) {

// Call the parent class constructor super(accountNumber,balance);

}

// Override the withdraw method from the parent class @Override

public void withdraw(double amount) {

// Check if the withdrawal would cause the balance to drop below $100 if (getBalance() - amount < 100) {

// Print a message if the minimum balance requirement is not met System.out.println("Minimum balance of $100 required!");

} else {

// Call the parent class withdraw method super.withdraw(amount);

}

}

}

public class Main {

public static void main(String[] args) {

// Print message to indicate creation of a BankAccount object

System.out.println("Create a Bank Account object (A/c No. BA1234) with initial balance of $500:");

// Create a BankAccount object (A/c No. "BA1234") with initial balance of $500 BankAccount BA1234 = new BankAccount("BA1234", 500);

// Print message to indicate deposit action

System.out.println("Deposit $1000 into account BA1234:");

// Deposit $1000 into account BA1234 BA1234.deposit(1000);

// Print the new balance after deposit

System.out.println("New balance after depositing $1000: $"+BA1234.getBalance());

// Print message to indicate withdrawal action

System.out.println("Withdraw $600 from account BA1234:");

// Withdraw $600 from account BA1234 BA1234.withdraw(600);

// Print the new balance after withdrawal

System.out.println("New balance after withdrawing $600: $" + BA1234.getBalance());

// Print message to indicate creation of another SavingsAccount object System.out.println("Create a SavingsAccount object (A/c No. SA1000) with initial

balance of $300:");

// Create a SavingsAccount object (A/c No. "SA1000") with initial balance of $300 SavingsAccount SA1000 = new SavingsAccount("SA1000", 300);

// Print message to indicate withdrawal action

System.out.println("Try to withdraw $250 from SA1000!");

// Withdraw $250 from SA1000 (balance falls below $100) SA1000.withdraw(250);

// Print the balance after attempting to withdraw $250

System.out.println("Balance after trying to withdraw $250: $" + SA1000.getBalance());

}

} OUTPUT:

