ASSIGNMENT:3 MID:M1095607

**Object Oriented Programming**

1. What are the objects and properties involved in the solution of below questions.
2. Find the number of blue cars running in Bengaluru?
3. Find the city where in the road, number of blue cars is maximum?

**Answer**

1. cars are the object so blue color is the property of the cars and the number of blue cars is showing the quantity of blue cars running in Bengaluru and bengaluru is the class as it is a specific city of India or showing specific class name.
2. city is the object since it specifies certain city name and road is the class name.Again, Blue color is the property of the cars, cars are the object and the numberof blue cars is showing the quantity of blue cars.
3. Justify if a car is a class or object

**Answer**

car is the name of the class from which the instance was created.

Example: Car car=new Car();

1. We have three options as
2. Tiger
3. White tiger
4. 2 year old white tiger named tipu

Which among these is a class and why?

**Answer**

**Tiger is the class** .

**White tiger** : color of the tiger is white.. so white color is the property of the tiger

**2 year old white tiger named tipu** : age of the tiger is 2 , color of the tiger is white and name is tipu. Name, Age and Color are found to be the properties of the class tiger.

1. What is the necessity of the reference variable

**Answer**

Reference variable is used to point object/values.

For example : Employee emp=new Employee;

Here emp is the reference variable.

1. when is the address of an object gets generated?

**Answer**

In Java when you are making an object from a class like Employee emp = new Employee();

emp is the address of a memory location which belongs to the type Employee .

1. Can we create a member variable called dog in a class called owner?

**Answer**

Yes we can create a member variable or the instance variable called dog in the class owner.

1. What is Encapsulation?

**Answer**

Encapsulation is the process of wrapping up of data variables and data methods together in a single unit.

By encapsulating a class variables,other classes cannot access them, and only the methods of the class can access them.

1. How to achieve Encapsulation?

**Answer**

Encapsulation can be achieved by Declaring all the variables in the class as private and writing public methods in the class to set and get the values of variables.

It is more defined with the setter and getter method.

1. What happens if getters and setters are made private?

**Answer**

Making the getters and setters private doesnt allow to access the methods from outside the class and so there is no use of getters and setters .

Therefore it is necessary to make it public so that they can be accessed outside the class.

1. What is the access specifiers associated with the constructor?

**Answer**

The access specifiers associated with the constructors are public,protected and private.

1. Can we achieve a job of a constructor using user defined method?

**Answer**

No it is not possible to use user defined methods instead of constructor because constructors are used to assign the values and instantiation for creating a object.

Eg: Employee emp=new Employee();

Here emp is used for creating a object and Employee() is the constructor which is used for instantiating a object in the class.

1. How to read properties of a dog object?

**Answer**

Properties in a class can be read by using the getters.

A class named dog has the following properties namely color , age.

These color and age can be accessed by getColor() and getAge() which are found to getters.

Dog d=new Dog(); d.setColor("White");d.setAge(2);

Now d.getColor() and d.getAge() are forund to be the getters which is used to read the properties of the dog object . (Output : White and 2).

1. How can we update the age of the dog from 2 to 3 years?

**Answer**

Inorder to update the dog object of property from age 2 to 3 ,

Dog d=new Dog(); d.setAge(3);

Now, d.getAge() will return 3.

Thus setters are used to update the values of the properties of the object in a class.

1. How can we delete a dog object?

**Answer**

In list , remove (Object) method is used to remove a object and is used if there exists array of objects.

It is also possible to assign properties with null value so that it will not be considered in the class.

So , remove(Dog d) can be used to remove a object from the array of objects with type Dog.

1. Can we create a dog object without using its constructor?

**Answer**

Creating a dog object with out initializing the constructor doesn't works and so it is necessary to initialize the dog object with the constructor.

For example: Dog d; d.getColor(); doesn't works as its not initialized while Dog d=new Dog(); and so d.getColor(); will work properly as the constructor is called or used.

16. A retail store wants to keep track of item id and item price of the five items sold by them. Based on the item purchased by the customer, item price must be identified and the computation of bill amount must be done as per the price and quantity of the item purchased. Write a program to implement the above scenario.

1.Represent the item with their ids and price in array.

2. Search for the item purchased by the customer in the item ids arrays and identify the respective price. Display and appropriate error message if the item is not found in the array.

3. If the item is found

a. Compute the bill amount as quantity purchased \* price identified.

b. Display the bill id, customer id, purchase id , quantity purchased, discount and bill amount.

4. Change the purchase item id value to 5006 and run the program again and observe the result.

**CODE:**

**package** com.mindtree.OOPS;

**import** java.util.\*;

**public** **class** RetailStore {

**public** **static** **void** main(String args[]) {

Scanner in=**new** Scanner(System.***in***);

Item[] items=**new** Item[5];

**for**(**int** i=0;i<5;i++) {

items[i]=**new** Item();

System.***out***.println("Enter the Price of item"+(i+1));

**int** price=in.nextInt();

System.***out***.println("Enter the Quantity of item"+(i+1));

**int** quantity=in.nextInt();

System.***out***.println("Enter the Discount of item"+(i+1));

**int** discount=in.nextInt();

items[i].setBillId(i+1);

items[i].setCustomerId(100+i);

items[i].setItemId(5000+i);

items[i].setPrice(price);

items[i].setQuantity(quantity);

items[i].setDiscount(discount);

}

System.***out***.println("Item Ids and Price");

**for**(**int** i=0;i<5;i++) {

System.***out***.println("Item Id: "+items[i].getItemId());

System.***out***.println("Price : "+items[i].getPrice());

System.***out***.println();

}

Scanner in4=**new** Scanner(System.***in***);

System.***out***.println("Enter the customer id to be searched");

**int** customer=in4.nextInt();

**int** found=0;

**for**(**int** j=0;j<items.length;j++) {

**if**(items[j].getCustomerId()==customer) {

found=1;

System.***out***.println("Price of the Item : "+items[j].getPrice());

System.***out***.println();

System.***out***.println("Bill Id: "+items[j].getBillId());

System.***out***.println("Customer Id: "+items[j].getCustomerId());

System.***out***.println("Purchase Id: "+items[j].getItemId());

System.***out***.println("Quantity Purchased: "+items[j].getQuantity());

System.***out***.println("Discount: "+items[j].getDiscount());

System.***out***.println("Bill Amount: "+(items[j].getPrice()\*items[j].getQuantity()));

System.***out***.println();

items[j].setItemId(5006);

System.***out***.println("Purchase Id changed to "+items[j].getItemId());

}

}

**if**(found==0) {

System.***out***.println("Item is not found in the array");

}

}

}

**class** Item{

**private** **int** billId;

**private** **int** customerId;

**private** **int** itemId;

**private** **int** price;

**private** **int** quantity;

**private** **int** discount;

**public** Item(**int** billId,**int** customerId,**int** itemId,**int** price,**int** quantity,**int** discount) {

**this**.billId=billId;

**this**.customerId=customerId;

**this**.itemId=itemId;

**this**.price=price;

**this**.quantity=quantity;

**this**.discount=discount;

}

**public** Item() {

**this**.billId=0;

**this**.customerId=0;

**this**.itemId=0;

**this**.price=0;

**this**.quantity=0;

**this**.discount=0;

}

**public** **int** getBillId() {

**return** billId;

}

**public** **void** setBillId(**int** billId) {

**this**.billId = billId;

}

**public** **int** getCustomerId() {

**return** customerId;

}

**public** **void** setCustomerId(**int** customerId) {

**this**.customerId = customerId;

}

**public** **int** getItemId() {

**return** itemId;

}

**public** **void** setItemId(**int** itemId) {

**this**.itemId = itemId;

}

**public** **int** getPrice() {

**return** price;

}

**public** **void** setPrice(**int** price) {

**this**.price = price;}

**public** **int** getQuantity() {

**return** quantity;

}

**public** **void** setQuantity(**int** quantity) {

**this**.quantity = quantity;

}

**public** **int** getDiscount() {

**return** discount;

}

**public** **void** setDiscount(**int** discount) {

**this**.discount = discount;

}

@Override

**public** String toString() {

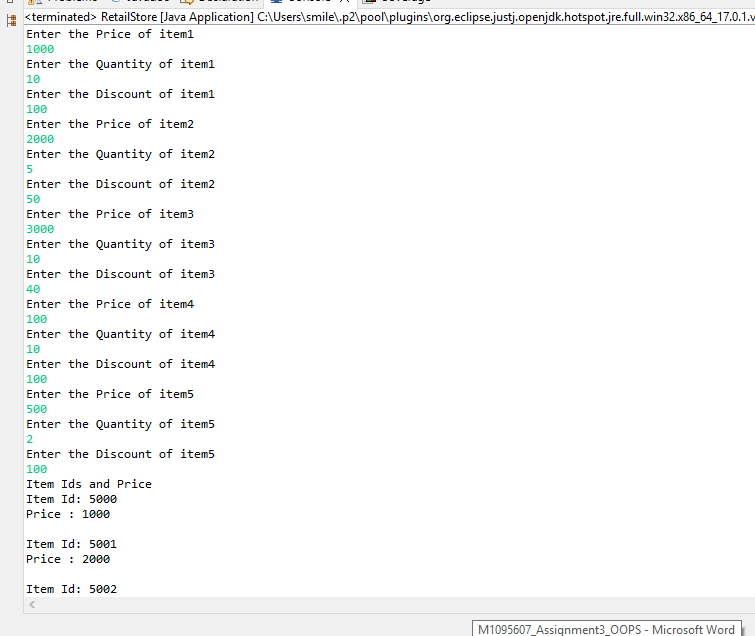
**return** "Item [billId=" + billId + ", customerId=" + customerId + ", itemId=" + itemId + ", price=" + price

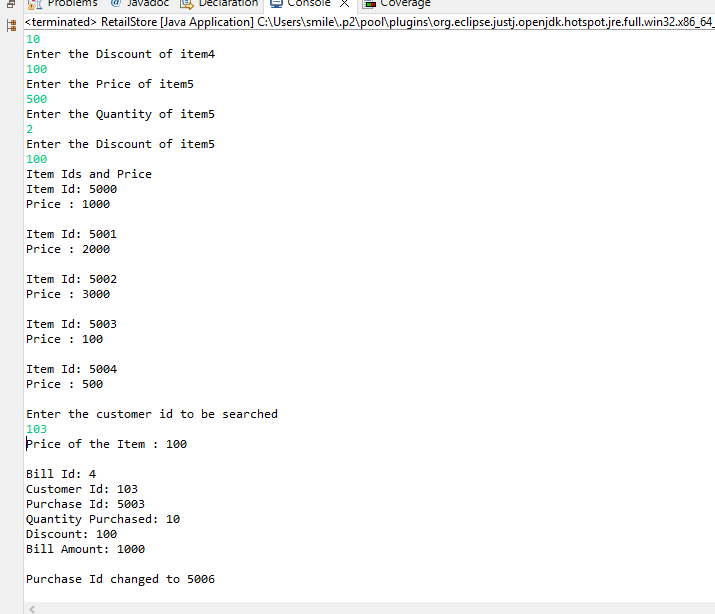
+ ", quantity=" + quantity + ", discount=" + discount + "]";

}

}

**OUTPUT:**

****

****

17. Create an Employee class with following attributes

Write a program which creates an instance of employee class and sets the values for all the attributes.

1. While setting values for empName,setEmpName() method should check for NullPointer and display appropriate error message.
2. While setting value for empDesig, the Designation must have any of the following values: developer,tester,Lead or Manager.If none of these values is matching,then setter method display ‘Invalid designation’ error message.
3. While setting value for empDept, the Department must have any of the following values: TTH,RCM,Digital,DevOps.If none of these values is matching , then setter method should display ‘Invalid Dept’ error message.

**CODE:**

**package** com.mindtree.OOPS;

**import** java.util.\*;

**public** **class** EmployeeMain {

**public** **static** **void** main(String[] args) {

Scanner in=**new** Scanner(System.***in***);

System.***out***.println("Enter the Employee ID");

**int** id=in.nextInt();

System.***out***.println("Enter the Employee Name");

String name=in.nextLine();

System.***out***.println("Enter the Employee Designation");

String empDesig=in.nextLine();

System.***out***.println("Enter the Employee Department");

String empDept=in.nextLine();

in.close();

Employee emp=**new** Employee();

emp.setempId(id);

emp.setEmpName(name);

emp.setEmpDesig(empDesig);

emp.setEmpDept(empDept);

**if**(emp.getEmpName().isEmpty()) {

System.***out***.println("Please Enter a Valid Name");

}

**else** **if**(emp.getEmpDept().isEmpty()) {

System.***out***.println("Please Enter a Valid Department");

}

**else** **if**(emp.getEmpDesig().isEmpty()) {

System.***out***.println("Please Enter a Valid Designation");

}

**else** {

System.***out***.println("Employee Details");

System.***out***.println("Employee Id : "+emp.getempId());

System.***out***.println("Employee Name : "+emp.getEmpName());

System.***out***.println("Employee Designation : "+emp.getEmpDesig());

System.***out***.println("Employee Department : "+emp.getEmpDept());

}

}

}

**class** Employee{

**private** **int** empId;

**private** String empName;

**private** String empDesig;

**private** String empDept;

**public** Employee() {

}

**public** Employee(**int** empId,String empName,String empDesig,String empDept) {

**this**.empId=empId;

**this**.empName=empName;

**this**.empDesig=empDesig;

**this**.empDept=empDept;

}

**public** **int** getempId() {

**return** empId;

}

**public** **void** setempId(**int** empId) {

**this**.empId = empId;

}

**public** String getEmpName() {

**return** empName;

}

**public** **void** setEmpName(String empName) {

**if**(empName!=**null**) {

**this**.empName = empName;

}

}

**public** String getEmpDesig() {

**return** empDesig;

}

**public** **void** setEmpDesig(String empDesig) {

**if**(empDesig.equals("Developer")||empDesig.equals("Lead")||empDesig.equals("Manager")||empDesig.equals("Tester")) {

**this**.empDesig = empDesig;

}

**else** {

**this**.empDesig="";

}

}

**public** String getEmpDept() {

**return** empDept;

}

**public** **void** setEmpDept(String empDept) {

**if**(empDept.equals("TTH")||empDept.equals("RCM")||empDept.equals("Digital")||empDept.equals("DevOps")) {

**this**.empDept=empDept;

}

**else** {

**this**.empDept="";

}

}

@Override

**public** String toString() {

**return** "Employee [empId=" + empId + ", empName=" + empName + ", empDesig=" + empDesig + ", empDept=" + empDept

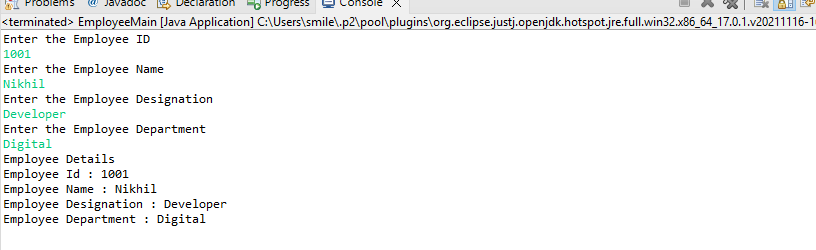
+ "]";

}

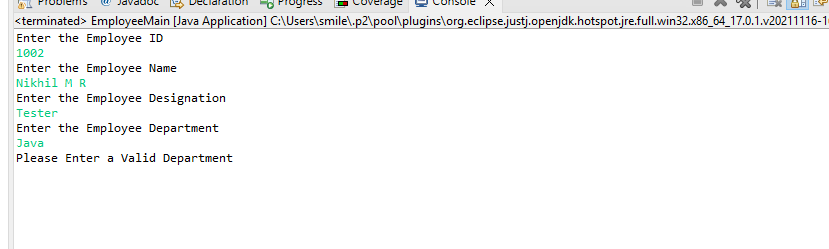
}

**OUTPUT:**

**a.**

****

**b.**

****

18. Develop a program that assists bookstore employees. For each book, the program should track the book’s title ,its price, its year of publication, and the author’s name. Develop an appropriate Java Class. Create instances of the class to represent these three books:

1. Daniel Defoe, Robinson Crusoe,$15.50,1719

2. Joseph Conrad, Heart of Darkness, $12.80,1902

3. Pat Conroy, Beach Music, $9.50,1996

**CODE:  
package** com.mindtree.OOPS;

**public** **class** BookStore {

**public** **static** **void** main(String[] args) {

Book book[]=**new** Book[3];

**for**(**int** i=0;i<3;i++) {

book[i]=**new** Book();

}

book[0]=**new** Book("Daniel Defoe","Robinson Crusoe","$15.50",1719);

book[1]=**new** Book("Joseph Conrad","Heart of Darkness","$12.80",1902);

book[2]=**new** Book("Pat Conroy","Beach Music","$9.50",1996);

System.***out***.println("Book Details");

System.***out***.println();

**for**(**int** i=0;i<3;i++) {

System.***out***.println("Book "+(i+1));

System.***out***.println("Book Name: "+book[i].getName());

System.***out***.println("Author of the Book: "+book[i].getAuthor());

System.***out***.println("Price of the Book: "+book[i].getPrice());

System.***out***.println("Year of Publication: "+book[i].getYear());

System.***out***.println();

}

}

}

**class** Book{

**private** String author;

**private** String name;

**private** String price;

**private** **int** year;

**public** Book() {

}

**public** Book(String author,String name,String price,**int** year) {

**this**.author=author;

**this**.name=name;

**this**.price=price;

**this**.year=year;

}

**public** String getAuthor() {

**return** author;

}

**public** **void** setAuthor(String author) {

**this**.author = author;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getPrice() {

**return** price;

}

**public** **void** setPrice(String price) {

**this**.price = price;

}

**public** **int** getYear() {

**return** year;

}

**public** **void** setYear(**int** year) {

**this**.year = year;

}

@Override

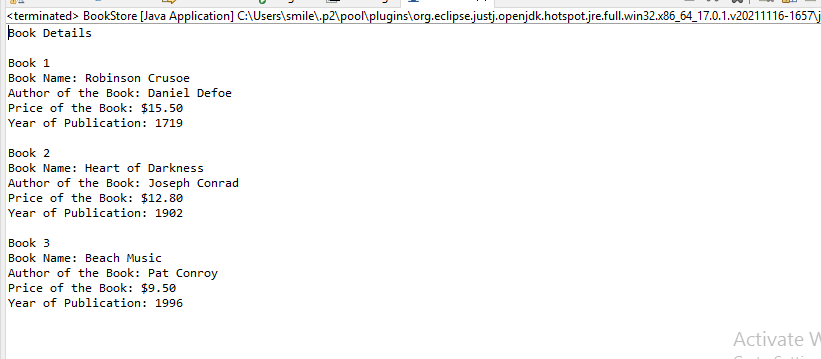
**public** String toString() {

**return** "Book [author=" + author + ", name=" + name + ", price=" + price + ", year=" + year + "]";

}

}

**OUTPUT:**

****

19. XYZ bank wants to maintain customer details.It will register the customer details whenever a person opens an account with the bank. Below is the customer class diagram:

At times, the customer registration process changes, here are the guidelines.

1. Admin may register customer by filling only ID, name and address details.
2. Admin may register customer by filling only ID and name.
3. Admin may register customer by filling all the details.

Write an application which implements above scenario .Write main method in separate class. Which creates different customer objects and invokes appropriate constructors, here is sample code:

Note: When other data members which are not initialized through constructors should have appropriate default values.

**CODE:**

**package** com.mindtree.OOPS;

**import** java.util.\*;

**public** **class** BankMain {

**public** **static** **void** main(String args[]) {

Scanner in=**new** Scanner(System.***in***);

Customer[] customer=**new** Customer[3];

**for**(**int** i=0;i<customer.length;i++) {

customer[i]=**new** Customer();

}

System.***out***.println("Enter the details for the customer");

System.***out***.println("Enter customer Id");

**int** id=in.nextInt();

System.***out***.println("Enter customer Name");

String name=in.nextLine();

System.***out***.println("Enter the Customer Address");

String address=in.nextLine();

System.***out***.println("Enter the customer account type");

String accType=in.nextLine();

System.***out***.println("Enter the Customer Balance");

**double** custBalance=in.nextDouble();

//Customers

customer[0]=**new** Customer(id,name,address); // Constructor with id,name and address

customer[1]=**new** Customer(id,name); // constructor with id and name

customer[2]=**new** Customer(id,name,address,accType,custBalance); //constructor with id,name,address,accType and custBalance

System.***out***.println("Bank details of the customer");

System.***out***.println();

**for**(**int** i=0;i<customer.length;i++) {

**if**(i==0) {System.***out***.println("Constructor with id,name and address");}**else** **if**(i==1) {System.***out***.println("constructor with id and name");}**else** {System.***out***.println("constructor with id,name,address,accType and custBalance");}

System.***out***.println("Customer ID: "+customer[i].getCustId());

System.***out***.println("Customer Name: "+customer[i].getCustName());

System.***out***.println("Customer Address: "+customer[i].getCustAddress());

System.***out***.println("Customer Account Type: "+customer[i].getAccType());

System.***out***.println("Customer Balance: "+customer[i].getCustBalance());

System.***out***.println();

}

}

}

**class** Customer{

**private** **int** custId;

**private** String custName;

**private** String custAddress;

**private** String accType;

**private** **double** custBalance;

**public** Customer() {

}

**public** Customer(**int** custId,String custName,String custAddress) {

**this**.custId=custId;

**this**.custName=custName;

**this**.custAddress=custAddress;

**this**.accType="Savings Account";

**this**.custBalance=1000;

}

**public** Customer(**int** custId,String custName) {

**this**.custId=custId;

**this**.custName=custName;

**this**.custAddress="Nagercoil";

**this**.accType="Savings Account";

**this**.custBalance=1000;

}

**public** Customer(**int** custId,String custName,String custAddress,String accType,**double** custBalance) {

**this**.custId=custId;

**this**.custName=custName;

**this**.custAddress=custAddress;

**this**.accType=accType;

**this**.custBalance=custBalance;

}

**public** **int** getCustId() {

**return** custId;

}

**public** **void** setCustId(**int** custId) {

**this**.custId = custId;

}

**public** String getCustName() {

**return** custName;

}

**public** **void** setCustName(String custName) {

**this**.custName = custName;

}

**public** String getCustAddress() {

**return** custAddress;

}

**public** **void** setCustAddress(String custAddress) {

**this**.custAddress = custAddress;

}

**public** String getAccType() {

**return** accType;

}

**public** **void** setAccType(String accType) {

**this**.accType = accType;

}

**public** **double** getCustBalance() {

**return** custBalance;

}

**public** **void** setCustBalance(**double** custBalance) {

**this**.custBalance = custBalance;

}

@Override

**public** String toString() {

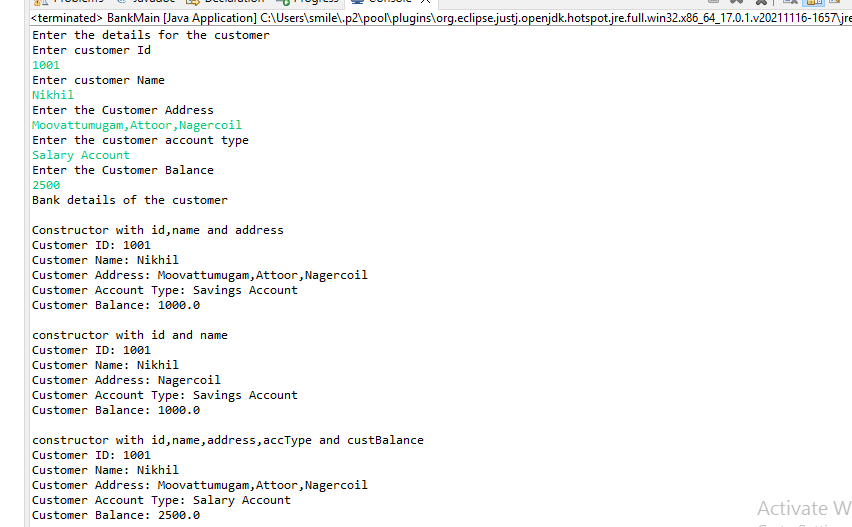
**return** "Customer [custId=" + custId + ", custName=" + custName + ", custAddress=" + custAddress + ", accType="

+ accType + ", custBalance=" + custBalance + "]";

}

}

**OUTPUT:**

****

20. Implement below given class diagram. Invoke constructor and methods of this class by creating appropriate object in main method.

Implement the withdraw(double amount) method: If amount is greater than balance then display error message; otherwise debit amount from balance and display the message “successfully withdrawn” +amount.

Implement the calculateInterest() method: calculation of simple interest for the balance maintained in the savings account.

**CODE:**

**package** com.mindtree.OOPS;

**import** java.util.\*;

**public** **class** AmountMain {

**public** **static** **void** main(String[] args) {

Scanner t=**new** Scanner(System.***in***);

System.***out***.println("Enter the Balance Rate: ");

**double** balance=t.nextDouble();

System.***out***.println("Enter the Interest Rate: ");

**int** interestRate=t.nextInt();

System.***out***.println("Enter the Account No: ");

**long** accountNo=t.nextLong();

SavingsAccount savings=**new** SavingsAccount(balance,interestRate,accountNo);

System.***out***.println("Details of Salary Account");

System.***out***.println("Balance rate : " +savings.getBalanceRate());

System.***out***.println("Interest Rate : "+savings.getInterestRate());

System.***out***.println("Account No : "+savings.getAccountNo());

System.***out***.println();

System.***out***.println("Enter the amount to be withdrawn");

**double** amount=t.nextDouble();

savings.withDraw(amount);

savings.calculateInterest();

}

}

**class** SavingsAccount{

**private** **double** balanceRate;

**private** **int** interestRate;

**private** **long** accountNo;

**public** SavingsAccount() {

}

**public** SavingsAccount(**double** balanceRate,**int** interestRate,**long** accountNo) {

**this**.balanceRate=balanceRate;

**this**.interestRate=interestRate;

**this**.accountNo=accountNo;

}

**public** **double** getBalanceRate() {

**return** balanceRate;

}

**public** **void** setBalanceRate(**double** balanceRate) {

**this**.balanceRate = balanceRate;

}

**public** **int** getInterestRate() {

**return** interestRate;

}

**public** **void** setInterestRate(**int** interestRate) {

**this**.interestRate = interestRate;

}

**public** **long** getAccountNo() {

**return** accountNo;

}

**public** **void** setAccountNo(**long** accountNo) {

**this**.accountNo = accountNo;

}

**public** **void** withDraw(**double** amount) {

**if**(amount>getBalanceRate()) {

System.***out***.println("Balance is less than the Entered Amount");

}

**else** {

**double** val=getBalanceRate()-amount;

System.***out***.println("Successfully Withdrawn : "+amount);

setBalanceRate(val);

System.***out***.println("Balance Rate After Withdrawal : "+getBalanceRate());

}

}

**public** **void** calculateInterest() {

**double** interest=(getBalanceRate()\*getInterestRate())/100;

System.***out***.println("Simple Interest : "+interest);

}

@Override

**public** String toString() {

**return** "SavingsAccount [balanceRate=" + balanceRate + ", interestRate=" + interestRate + ", accountNo="

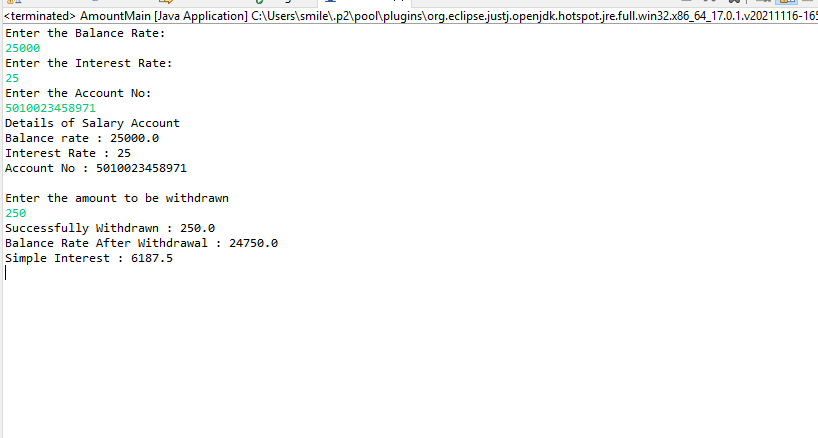
+ accountNo + "]";

}

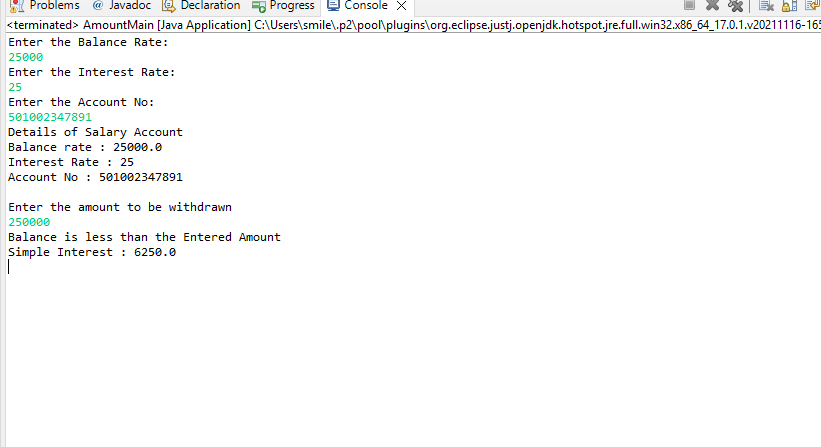
}

**OUTPUT:**

**a.**

****

**b.**

****

21. A coffee shop would like to find out the customer feedback rating about its services.

Example : Assume that the shop will collect feedback from ‘N’ customers. Following are the sample customer feedback values.

Customer 1:3 out of 5

Customer 2:4 out of 5

Customer 3:2.5 out of 5

Write an application which creates array of ‘N; customer objects to store feedback values of these.

**CODE:**

**package** com.mindtree.OOPS;

**import** java.util.\*;

**public** **class** CoffeeShop {

**public** **static** **void** main(String args[]) {

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Enter the number of customers");

**int** n=scan.nextInt();

Customers[] customer=**new** Customers[n];

**for**(**int** i=0;i<n;i++) {

customer[i]=**new** Customers();

System.***out***.println("Enter the customer name");

String name=scan.nextLine();

System.***out***.println("Enter the Customer Mobile Number");

String mobileNo=scan.nextLine();

System.***out***.println("Enter yoour Feedback Rating out of 5");

**double** feedBackRating=scan.nextDouble();

customer[i].setName(name);

customer[i].setMobileNo(mobileNo);

customer[i].setFeedBackRating(feedBackRating);

}

System.***out***.println("Feedback by Customers");

System.***out***.println();

**for**(**int** i=0;i<n;i++) {

System.***out***.println("Customer :"+(i+1));

System.***out***.println("Name of the customer: "+customer[i].getName());

System.***out***.println("Mobile Number of the Customer: "+customer[i].getMobileNo());

System.***out***.println("Feedback Rating : "+customer[i].getFeedBackRating());

System.***out***.println();

}

}

}

**class** Customers{

**private** String name;

**private** String mobileNo;

**private** **double** feedBackRating;

**public** Customers() {

}

**public** Customers(String name,String mobileNo,**double** feedbackRating) {

**this**.name=name;

**this**.mobileNo=mobileNo;

**this**.feedBackRating=feedBackRating;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getMobileNo() {

**return** mobileNo;

}

**public** **void** setMobileNo(String mobileNo) {

**this**.mobileNo = mobileNo;

}

**public** **double** getFeedBackRating() {

**return** feedBackRating;

}

**public** **void** setFeedBackRating(**double** feedBackRating) {

**this**.feedBackRating = feedBackRating;

}

@Override

**public** String toString() {

**return** "Customers [name=" + name + ", mobileNo=" + mobileNo + ", feedBackRating=" + feedBackRating + "]";

}

}

**OUTPUT:**

****

22.Analyze the class diagram

a. Identify the classes.

b. Identify the relationships between each classes.

c. Identify the attributes and methods.

**Answer:**

1. **Classes**

**Company**

**Employee**

**Manager**

**Contractor**

1. **Relationships between the classes**

**Company to Employee - Association and Multiplicity**

**Employee to Company- Aggregation**

**Manager to Employee-Inheritance**

**Employee to Manager-Association and Multiplicity**

**Contractor to Manager-Inheritance**

1. **Attributes and Methods**

**Company Attributes**

**\*name**

**\*Employees**

**Company Methods**

**\*getName()**

**\*getEmployees()**

**Employee Attributes**

**\*name**

**\*employeeNumber**

**\* salary**

**\* Manager**

**Employee Methods**

**\*getName()**

**\*getEmployeeNumber()**

**\*getSalary()**

**\* getManager()**

**Manager**

**Manager Attributes**

**\*manages**

**Manager Methods**

**\*addTeamMembers()**

**\*getTeamMembers()**

**Contractor**

**Contractor Attributes**

**\*lengthOfContract**

**Contractor Methods**

**\*getLengthOfContract()**

23. Given a class diagram for a problem, use the method and constructor overloading concepts to solve the problem and test using a set of valuesin an IDE.

Problem Description: The admin of a pre-university colleges wants to calculate the marks of students in a particular course based on same criteria. Write a Java program to implement the below given class diagram.

Starter class

Constructor: Initializes Student object withstudentId, studentName and secondChance.

**identifyMarks(float) method**:This method is used to set the marks of the student if the student has cleared in the first chance itself, i.e. second chance is false. This method accepts the marksscored by the student which must be set in the marks instance variable.

**identifyMarks (float, float) method**: This method is used to set the marks of the student if the student has taken the second chance i.e. second chance is true. This method accepts the marks scored by the student in the first chance and second chance. The maximum of both these marks must be identified and set in the marks instance variable.

**Starter Class:**

Write a starter class named StudentApp.

Step1: Create an object of Student class by passing appropriate values to theconstructor.

Step2: Based on the value used for second chance instance variable,invoke the appropriate identifyMarks() method.

Step3: Invoke the getter methods and display all the instance variable values of theStudent object created. Create one more object (use different value for secondchance) by repeating steps 1 to 3 and test your program.

**CODE:**

**package** com.mindtree.OOPS;

**import** java.util.\*;

**public** **class** StudentApp {

**public** **static** **void** main(String args[]) {

Scanner scan=**new** Scanner(System.***in***);

Student[] student=**new** Student[2];

**for**(**int** i=0;i<student.length;i++) {

System.***out***.println("Enter the Student Id");

**int** id=scan.nextInt();

System.***out***.println("Enter the Student Name");

String name=scan.nextLine();

System.***out***.println("Second Chance");

Boolean chance=scan.nextBoolean();

student[i]=**new** Student(id,name,chance);

**if**(chance==**true**) {

System.***out***.println("Enter the mark1");

**float** mark1=scan.nextFloat();

System.***out***.println("Enter the mark2");

**float** mark2=scan.nextFloat();

student[i].identifyMarks(mark1, mark2);

}

**else** {

System.***out***.println("Enter the marks");

**float** marks=scan.nextFloat();

student[i].identifyMarks(marks);

}

System.***out***.println();

}

System.***out***.println("Students Details");

**for**(**int** i=0;i<student.length;i++) {

System.***out***.println("Student ID : "+student[i].getStudentId());

System.***out***.println("Student Name : "+student[i].getStudentName());

System.***out***.println("Marks : "+student[i].getMarks());

System.***out***.println("Second Chance : "+student[i].isSecondChance());

System.***out***.println();

}

}

}

**class** Student{

**private** **int** studentId;

**private** String studentName;

**private** **float** marks;

**private** **boolean** secondChance;

**public** Student() {

}

**public** Student(**int** studentId,String studentName,**boolean** secondChance) {

**this**.studentId=studentId;

**this**.studentName=studentName;

**this**.secondChance=secondChance;

}

**public** **int** getStudentId() {

**return** studentId;

}

**public** **void** setStudentId(**int** studentId) {

**this**.studentId = studentId;

}

**public** String getStudentName() {

**return** studentName;

}

**public** **void** setStudentName(String studentName) {

**this**.studentName = studentName;

}

**public** **float** getMarks() {

**return** marks;

}

**public** **void** setMarks(**float** marks) {

**this**.marks = marks;

}

**public** **boolean** isSecondChance() {

**return** secondChance;

}

**public** **void** setSecondChance(**boolean** secondChance) {

**this**.secondChance = secondChance;

}

**public** **void** identifyMarks(**float** mark) {

**this**.setMarks(mark);

}

**public** **void** identifyMarks(**float** mark1,**float** mark2) {

**if**(mark1>mark2) {

**this**.setMarks(mark1);

}

**else** {

**this**.setMarks(mark2);

}

}

@Override

**public** String toString() {

**return** "Student [studentId=" + studentId + ", studentName=" + studentName + ", marks=" + marks

+ ", secondChance=" + secondChance + "]";

}

}

**OUTPUT:**

****

24. Create menu driven program to implement following scenario:

1. Create Student Record

2. Display Student Names in sorted order based on branch (alphabetical order)

3.Display Student ID in ascending sorted order

Each Student object should have following values:

• ID

• Name

• Branch

• Score

**CODE:**

**import** java.util.\*;

**public** **class** StudentRec {

**public** **static** **void** main(String args[]) {

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Enter the number of student records");

**int** n=scan.nextInt();

Student students[]=**new** Student[n];

**int** index=0;

**boolean** cond=**true**;

**do** {

System.***out***.println("1. Create Student Record");

System.***out***.println("2. Display Student Names in sorted order based on branch (alphabetical order) ");

System.***out***.println("3. Display Student ID in ascending sorted order");

System.***out***.println("4. Exit");

System.***out***.println("Enter the Option");

**int** option=scan.nextInt();

**switch**(option) {

**case** 1:

System.***out***.println();

*addStudentRecord*(students,index);

System.***out***.println("Record Added");

index++;

**break**;

**case** 2:

System.***out***.println();

System.***out***.println("Student Names in sorted order based on branch (alphabetical order) ");

*namesSortedOrder*(students);

**break**;

**case** 3:

System.***out***.println();

System.***out***.println("Student ID in ascending sorted order");

**int**[] idVal=*studentSortedOrder*(students);

**for**(**int** i=0;i<idVal.length;i++) {

System.***out***.println(idVal[i]);

}

**break**;

**case** 4:

cond=**false**;

System.***out***.println("Exiting");

**break**;

**default**:

System.***out***.println("Please Enter a Valid Option");

**break**;

}

}**while**(cond);

}

**public** **static** **void** addStudentRecord(Student[] students,**int** index) {

Scanner input=**new** Scanner(System.***in***);

System.***out***.println("Enter the Student Id");

**int** id=input.nextInt();

System.***out***.println("Enter the Student Name");

String name=input.nextLine();

System.***out***.println("Enter the Student Branch");

String branch=input.nextLine();

branch=branch.toLowerCase();

System.***out***.println("Enter the Student Score");

**float** score=input.nextFloat();

Student newStu=**new** Student(id,name,branch,score);

students[index]=newStu;

}

**public** **static** **void** namesSortedOrder(Student[] students) {

**for**(**int** i=0;i<students.length;i++) {

**for**(**int** j=i+1;j<students.length;j++) {

**if**(students[i].getBranch().compareTo(students[j].getBranch())>0) {

Student temp=students[i];

students[i]=students[j];

students[j]=temp;

}

}

}

**for**(**int** i=0;i<students.length;i++) {

System.***out***.println("Name : "+students[i].getName()+" - ("+students[i].getBranch()+")");

}

}

**public** **static** **int**[] studentSortedOrder(Student[] students){

**int**[] sortedId=**new** **int**[students.length];

**for**(**int** i=0;i<students.length;i++) {

**for**(**int** j=i+1;j<students.length;j++) {

**if**(students[i].getId()>students[j].getId()) {

Student temp=students[i];

students[i]=students[j];

students[j]=temp;

}

}

}

**for**(**int** i=0;i<students.length;i++) {

sortedId[i]=students[i].getId();

}

**return** sortedId;

}

}

**class** Student{

**private** **int** id;

**private** String name;

**private** String branch;

**private** **float** score;

**public** Student() {

}

**public** Student(**int** id,String name,String branch,**float** score) {

**this**.id=id;

**this**.name=name;

**this**.branch=branch;

**this**.score=score;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getBranch() {

**return** branch;

}

**public** **void** setBranch(String branch) {

**this**.branch = branch;

}

**public** **float** getScore() {

**return** score;

}

**public** **void** setScore(**float** score) {

**this**.score = score;

}

@Override

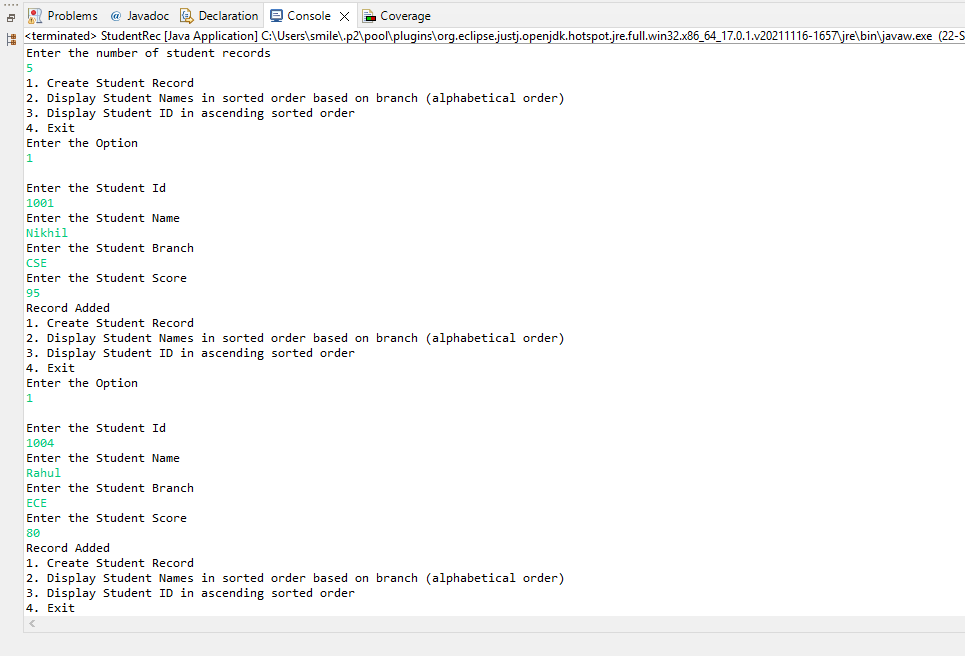
**public** String toString() {

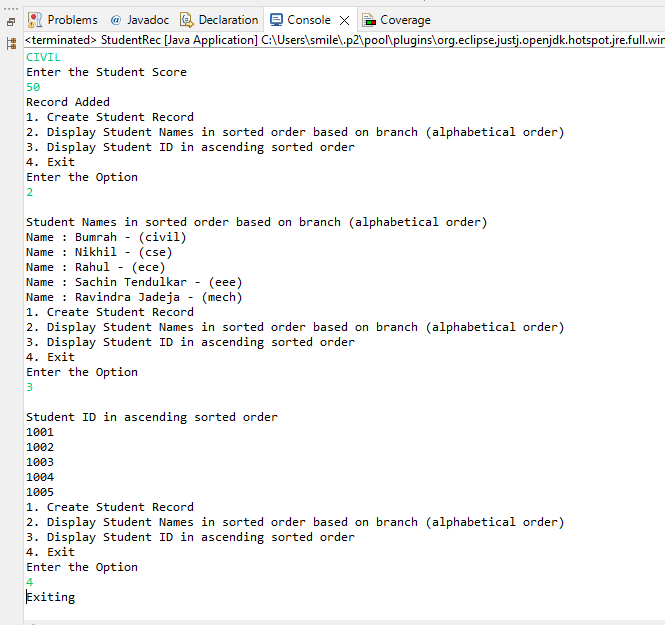
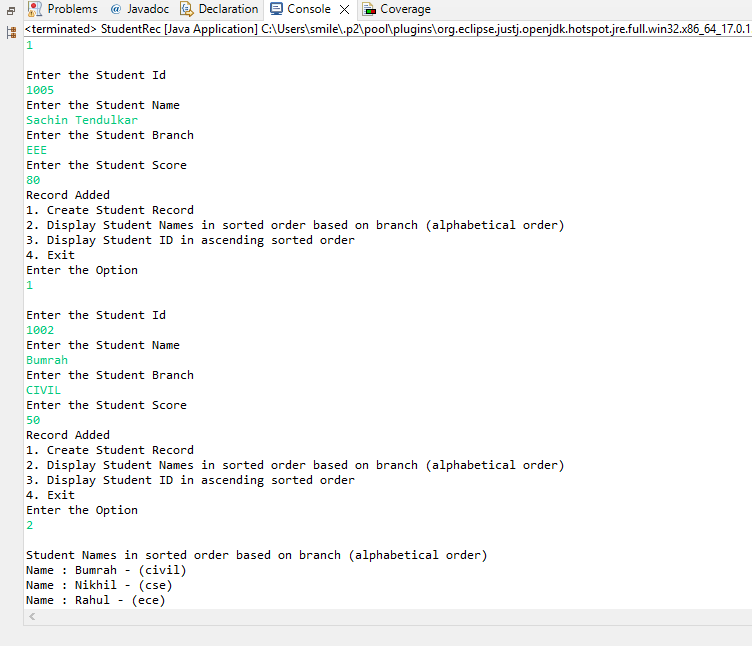
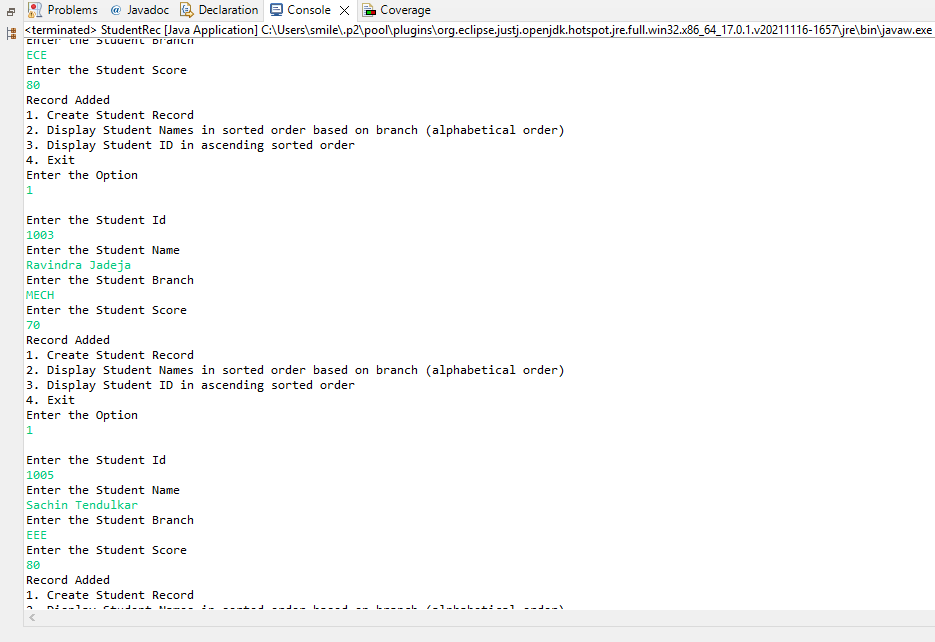
**return** "Student [id=" + id + ", name=" + name + ", branch=" + branch + ", score=" + score + "]";

}

}

**OUTPUT:**

****

****

25. Create a method which accepts array of ‘Student’ objects and returns Student Object who has scored highest marks.

Note:Each student object should have following values.

a. ID

b. Name

c. Branch

d. Score

**CODE:**

**package** com.mindtree.OOP;

**import** java.util.\*;

**public** **class** StudentMain {

**public** **static** **void** main(String[] args) {

Scanner newScan=**new** Scanner(System.***in***);

System.***out***.println("Enter the number of students");

**int** n=newScan.nextInt();

Student[] student=**new** Student[n];

**for**(**int** i=0;i<n;i++) {

System.***out***.println("Enter the Student Id");

**int** id=newScan.nextInt();

System.***out***.println("Enter the Student Name");

String name=newScan.nextLine();

System.***out***.println("Enter the Student Branch");

String branch=newScan.nextLine();

System.***out***.println("Enter the Student Score");

**float** score=newScan.nextFloat();

student[i]=**new** Student(id,name,branch,score);

}

Student highestStudent=*getStudentDetails*(student);

System.***out***.println("Student Details With Highest Score ");

System.***out***.println("Student ID: "+highestStudent.getId());

System.***out***.println("Student Name: "+highestStudent.getName());

System.***out***.println("Student Branch: "+highestStudent.getBranch());

System.***out***.println("Student Score: "+highestStudent.getScore());

}

**public** **static** Student getStudentDetails(Student[] students) {

**float** max=0;Student highestStudent=**new** Student();

**for**(Student student: students ) {

**if**(student.getScore()>max) {

max=student.getScore();

highestStudent=student;

}

}

**return** highestStudent;

}

}

**class** Student{

**private** **int** id;

**private** String name;

**private** String branch;

**private** **float** score;

**public** Student() {

}

**public** Student(**int** id,String name,String branch,**float** score) {

**this**.id=id;

**this**.name=name;

**this**.branch=branch;

**this**.score=score;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getBranch() {

**return** branch;

}

**public** **void** setBranch(String branch) {

**this**.branch = branch;

}

**public** **float** getScore() {

**return** score;

}

**public** **void** setScore(**float** score) {

**this**.score = score;

}

@Override

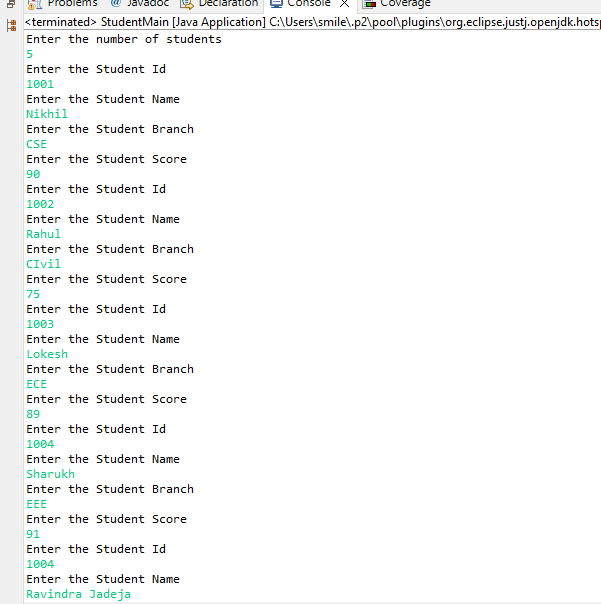
**public** String toString() {

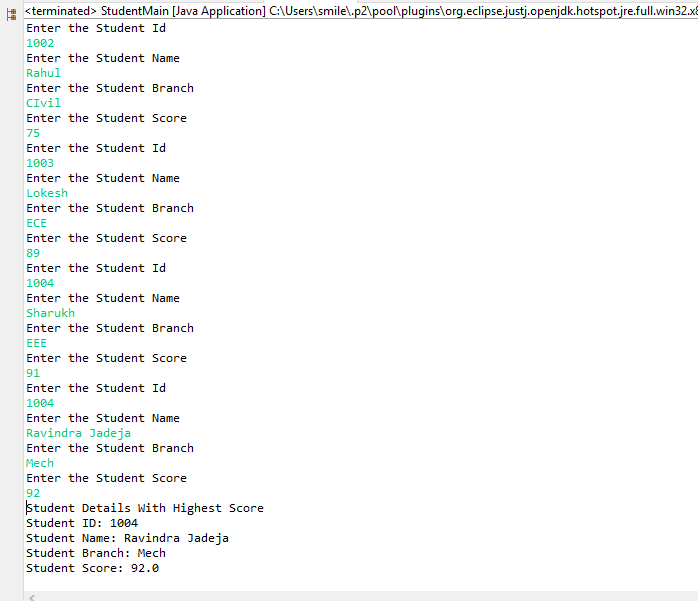
**return** "Student [id=" + id + ", name=" + name + ", branch=" + branch + ", score=" + score + "]";

}

}

**OUTPUT:**

****

****

26.Write a menu driven program to implement a solution to manage customers of a bank with following options:

1. **Create bank user**

a. Every bank user has following details: account number,name,account type[saving,loan and current], Date of birth,balance and address

2**. Update user details**

a. This option allows user to update either one of the following details: address and name

3. **Delete user details**

a.This option to delete user details from the record

4. **Display user details**

a.Display user details based on ID

b. Display user details based on Name

c. Display all users who have non-zero balance

d. Display user sorted based on address (location wise)

e. Display users sorted based on account id

5. **Exit**

**CODE:**

**package** com.mindtree.ObjectOrientedProg;

**import** java.util.\*;

**public** **class** BankRec {

**public** **static** **void** main(String args[]) {

Scanner in=**new** Scanner(System.***in***);

System.***out***.println("Enter the number of Bank Records");

**int** n=in.nextInt();

**int** index=0;

**boolean** cond=**true**;

Bank[] bank=**new** Bank[n];

**do** {

System.***out***.println("1. Create bank user");

System.***out***.println("2. Update user details");

System.***out***.println("3. Delete user details ");

System.***out***.println("4. Display user details");

System.***out***.println("5. Exit");

System.***out***.println();

System.***out***.println("Enter the option");

**int** option=in.nextInt();

**switch**(option) {

**case** 1:

System.***out***.println();

**if**(index==bank.length && index<=n) {

Bank[] existBank=**new** Bank[bank.length+1];

**for**(**int** i=0;i<existBank.length-1;i++) {

existBank[i]=bank[i];

}

existBank[index]=*createBankAcc*();

System.***out***.println(existBank[index]);

bank=**new** Bank[existBank.length];

**for**(**int** i=0;i<bank.length;i++) {

bank[i]=existBank[i];

}

}

**else** {

bank[index]=*createBankAcc*();

System.***out***.println();

}

index++;

**break**;

**case** 2:

System.***out***.println();

*updateUserDetails*(bank);

System.***out***.println();

**break**;

**case** 3:

System.***out***.println();

Bank[] updatedBank=*deleteUserDetails*(bank);

bank=**new** Bank[updatedBank.length];

**for**(**int** i=0;i<updatedBank.length;i++) {

bank[i]=updatedBank[i];

}

System.***out***.println();

**break**;

**case** 4:

System.***out***.println("1. Display user details based on Id");

System.***out***.println("2. User Details based on Name");

System.***out***.println("3. Display all users who have non-zero balance");

System.***out***.println("4.Display user sorted based on address ");

System.***out***.println("5. Display users sorted based on account id");

Scanner subSwitch=**new** Scanner(System.***in***);

System.***out***.println("Enter the option ");

**int** opt=subSwitch.nextInt();

**switch**(opt) {

**case** 1:

System.***out***.println();

*displayId*(bank);

System.***out***.println();

**break**;

**case** 2:

System.***out***.println();

*displayName*(bank);

System.***out***.println();

**break**;

**case** 3:

System.***out***.println();

*displayBalance*(bank);

System.***out***.println();

**break**;

**case** 4:

System.***out***.println();

*displayAddress*(bank);

System.***out***.println();

**break**;

**case** 5:

System.***out***.println();

*displayAccount*(bank);

System.***out***.println();

**break**;

**default**:

System.***out***.println("Enter a valid option");

**break**;

}

**break**;

**case** 5:

cond=**false**;

System.***out***.println("Exitting");

**break**;

**default**:

System.***out***.println("Enter a valid option");

**break**;

}

}**while**(cond);

}

**public** **static** Bank createBankAcc() {

Scanner in=**new** Scanner(System.***in***);

System.***out***.println("Enter Id");

**int** id=in.nextInt();

System.***out***.println("Enter name");

String name=in.nextLine();

System.***out***.println("Enter account number");

String accountNo=in.nextLine();

System.***out***.println("Enter the account type");

String accountType=in.nextLine();

System.***out***.println("Enter the date of birth");

String date=in.nextLine();

System.***out***.println("Enter the balance");

**long** balance=in.nextLong();

System.***out***.println("Enter the address");

String address=in.nextLine();

Bank addBank=**new** Bank(id,accountNo,name,accountType,date,balance,address);

System.***out***.println("Bank user added");

**return** addBank;

}

**public** **static** **void** updateUserDetails(Bank[] banks) {

Scanner input=**new** Scanner(System.***in***);

System.***out***.println("Enter the account number to update");

String accNo=input.nextLine();

**int** i=0;

**for**(Bank bank:banks) {

**if**(bank.getAccountNo().equals(accNo)) {

i=1;

System.***out***.println("Enter the Updated Name");

String name=input.nextLine();

System.***out***.println("Enter the updated Address");

String address=input.nextLine();

bank.setName(name);

bank.setAddress(address);

System.***out***.println("Bank Details Updated Successfully");

}

}

**if**(i==0) {

System.***out***.println("No such Account Number Found");

}

}

**public** **static** Bank[] deleteUserDetails(Bank[] banks) {

Scanner input=**new** Scanner(System.***in***);

Bank[] tempBank=**new** Bank[banks.length];

**for**(**int** i=0;i<tempBank.length;i++) {

tempBank[i]=banks[i];

}

System.***out***.println("Enter the Bank Account Number to Delete");

String accNo=input.nextLine();

**int** index=-1;

**for**(**int** i=0;i<tempBank.length;i++) {

**if**(tempBank[i].getAccountNo().equals(accNo)) {

index=i;

**for**(**int** j=index;j<tempBank.length-1;j++) {

tempBank[j]=tempBank[j+1];

}

System.***out***.println("Account Number: "+accNo+" deleted");

}

}

**if**(index==-1) {

System.***out***.println("No such account number found");

}

**else** {

tempBank[tempBank.length-1].setRemoved(**true**);

banks=**new** Bank[tempBank.length-1];

**for**(**int** i=0;i<banks.length;i++) {

banks[i]=tempBank[i];

}

}

**return** banks;

}

**public** **static** **void** displayId(Bank[] banks) {

System.***out***.println("Enter the Bank User ID");

Scanner in=**new** Scanner(System.***in***);

**int** id=in.nextInt();

**int** i=0;

**for**(Bank bank: banks) {

**if**(bank.getId()==id) {

System.***out***.println("Account Number : "+bank.getAccountNo());

System.***out***.println("Name : "+bank.getName());

System.***out***.println("Account Type : "+bank.getAccountType());

System.***out***.println("Date of birth : "+bank.getDateOfBirth());

System.***out***.println("Balance : "+bank.getBalance());

System.***out***.println("Address : "+bank.getAddress());

i=1;

}

}

**if**(i==0) {

System.***out***.println("No such Bank ID");

}

}

**public** **static** **void** displayName(Bank[] banks) {

System.***out***.println("Enter the user name");

Scanner input=**new** Scanner(System.***in***);

String name=input.nextLine();

**int** j=0;

**for**(Bank bank: banks) {

**if**(bank.getName().equals(name)) {

System.***out***.println("Account Number : "+bank.getAccountNo());

System.***out***.println("Name : "+bank.getName());

System.***out***.println("Account Type : "+bank.getAccountType());

System.***out***.println("Date of birth : "+bank.getDateOfBirth());

System.***out***.println("Balance : "+bank.getBalance());

System.***out***.println("Address : "+bank.getAddress());

j=1;

}

}

**if**(j==0) {

System.***out***.println("No such Bank User found");

}

}

**public** **static** **void** displayBalance(Bank[] banks) {

System.***out***.println("All users who have non-zero balance");

**for**(Bank bank: banks) {

**if**(bank.getBalance()>0) {

System.***out***.println("Account Number : "+bank.getAccountNo());

System.***out***.println("Name : "+bank.getName());

System.***out***.println("Account Type : "+bank.getAccountType());

System.***out***.println("Date of birth : "+bank.getDateOfBirth());

System.***out***.println("Balance : "+bank.getBalance());

System.***out***.println("Address : "+bank.getAddress());

}

}

}

**public** **static** **void** displayAddress(Bank[] banks) {

System.***out***.println("User sorted based on address ");

**for**(**int** k=0;k<banks.length;k++) {

**for**(**int** l=k+1;l<banks.length;l++) {

**if**(banks[k].getAddress().compareTo(banks[l].getAddress())>0) {

Bank temp=banks[k];

banks[k] = banks[l];

banks[l]= temp;

}

}

}

**for**(Bank bank:banks) {

System.***out***.println("Account Number : "+bank.getAccountNo());

System.***out***.println("Name : "+bank.getName());

System.***out***.println("Account Type : "+bank.getAccountType());

System.***out***.println("Date of birth : "+bank.getDateOfBirth());

System.***out***.println("Balance : "+bank.getBalance());

System.***out***.println("Address : "+bank.getAddress());

System.***out***.println();

}

}

**public** **static** **void** displayAccount(Bank[] banks) {

System.***out***.println("Users sorted based on account id");

**for**(**int** k=0;k<banks.length;k++) {

**for**(**int** l=k+1;l<banks.length;l++) {

**if**(banks[k].getId()>banks[l].getId()) {

Bank temp=banks[k];

banks[k] = banks[l];

banks[l]= temp;

}

}

}

**for**(Bank bank:banks) {

System.***out***.println("Account Number : "+bank.getAccountNo());

System.***out***.println("Name : "+bank.getName());

System.***out***.println("Account Type : "+bank.getAccountType());

System.***out***.println("Date of birth : "+bank.getDateOfBirth());

System.***out***.println("Balance : "+bank.getBalance());

System.***out***.println("Address : "+bank.getAddress());

System.***out***.println();

}

}

}

**class** Bank{

**private** **int** id;

**private** String accountNo;

**private** String name;

**private** String accountType;

**private** String dateOfBirth;

**private** **long** balance;

**private** String address;

**private** **boolean** isRemoved;

**public** Bank() {

}

**public** Bank(**int** id,String accountNo,String name,String accountType,String dateOfBirth,**long** balance,String address) {

**this**.id=id;

**this**.accountNo=accountNo;

**this**.name=name;

**this**.accountType=accountType;

**this**.dateOfBirth=dateOfBirth;

**this**.balance=balance;

**this**.address=address;

**this**.isRemoved=**false**;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getAccountNo() {

**return** accountNo;

}

**public** **void** setAccountNo(String accountNo) {

**this**.accountNo = accountNo;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getAccountType() {

**return** accountType;

}

**public** **void** setAccountType(String accountType) {

**this**.accountType = accountType;

}

**public** String getDateOfBirth() {

**return** dateOfBirth;

}

**public** **void** setDateOfBirth(String dateOfBirth) {

**this**.dateOfBirth = dateOfBirth;

}

**public** **long** getBalance() {

**return** balance;

}

**public** **void** setBalance(**long** balance) {

**this**.balance = balance;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** **boolean** isRemoved() {

**return** isRemoved;

}

**public** **void** setRemoved(**boolean** isRemoved) {

**this**.isRemoved = isRemoved;

}

@Override

**public** String toString() {

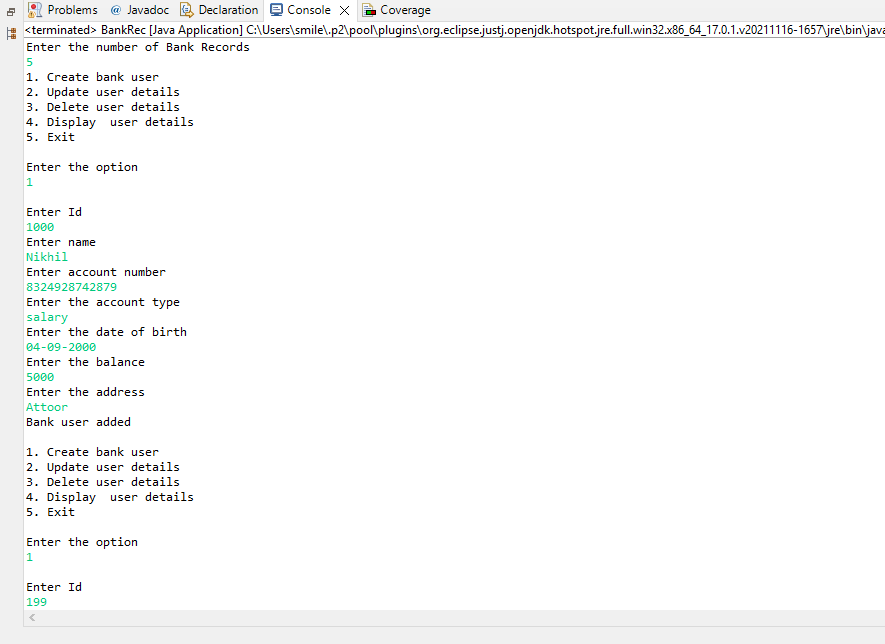
**return** "Bank [id=" + id + ", accountNo=" + accountNo + ", name=" + name + ", accountType=" + accountType

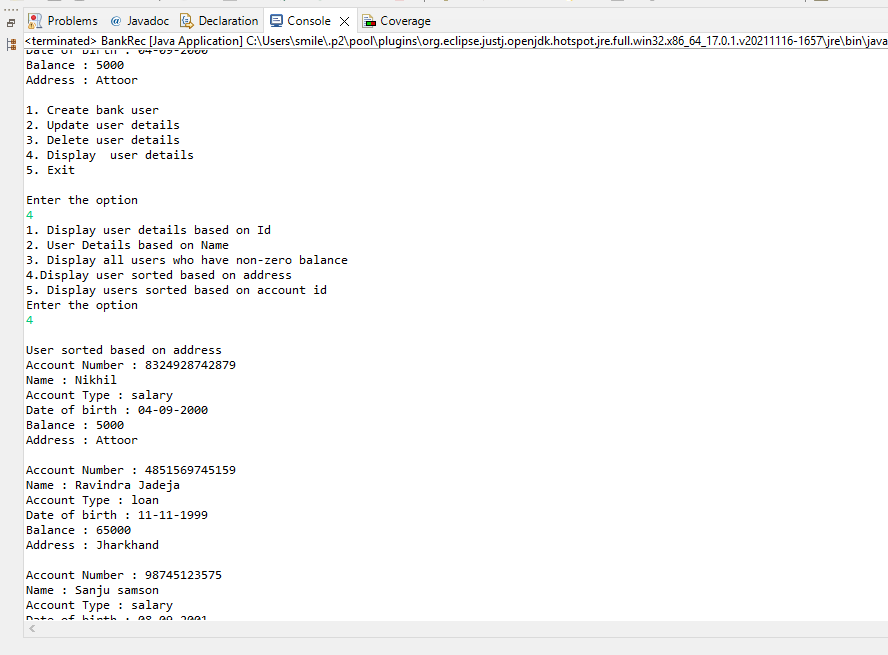
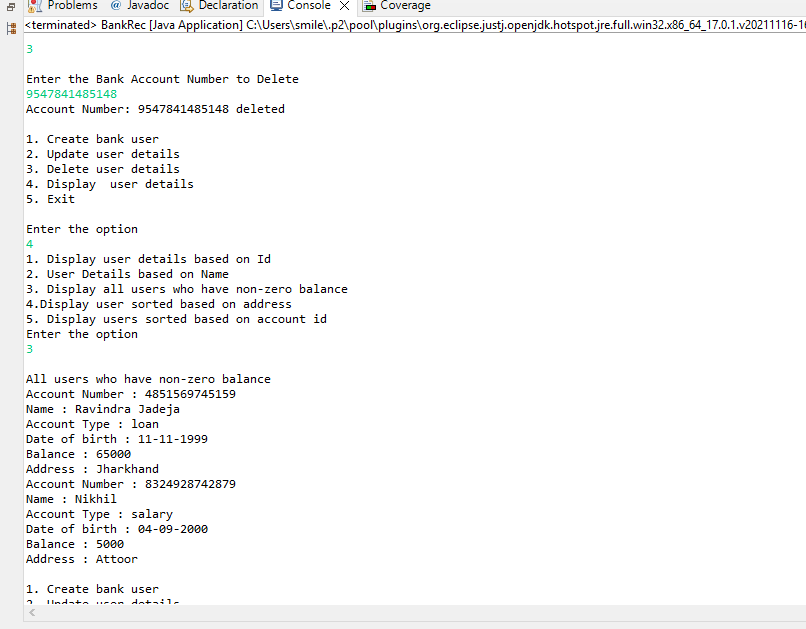
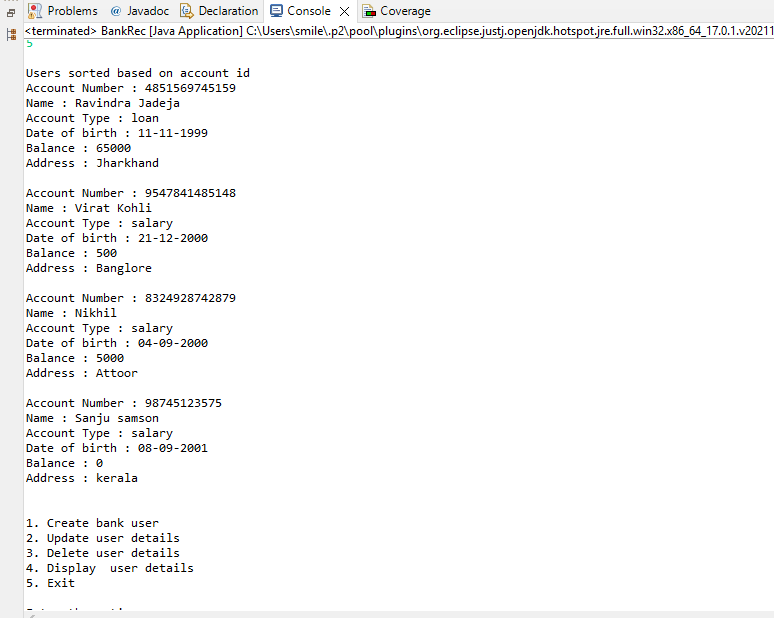
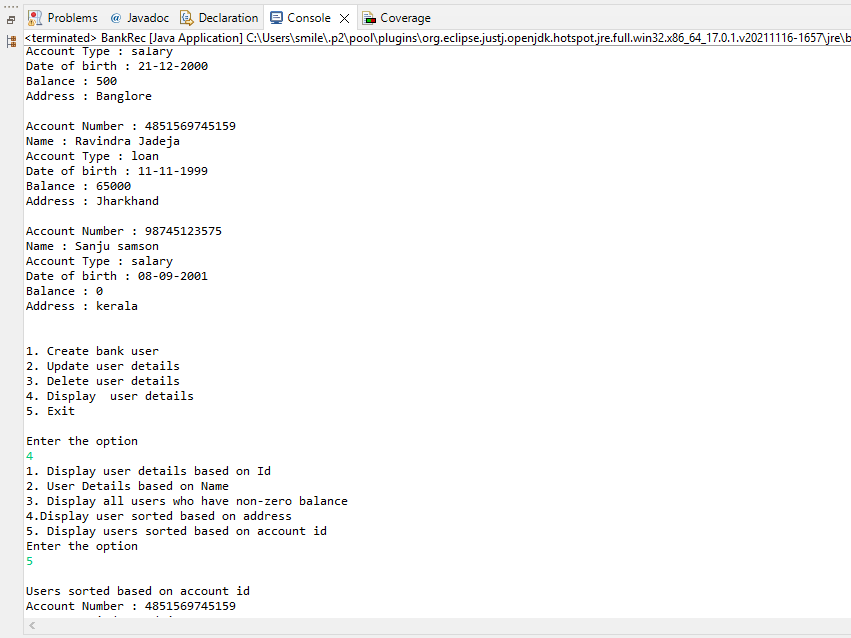
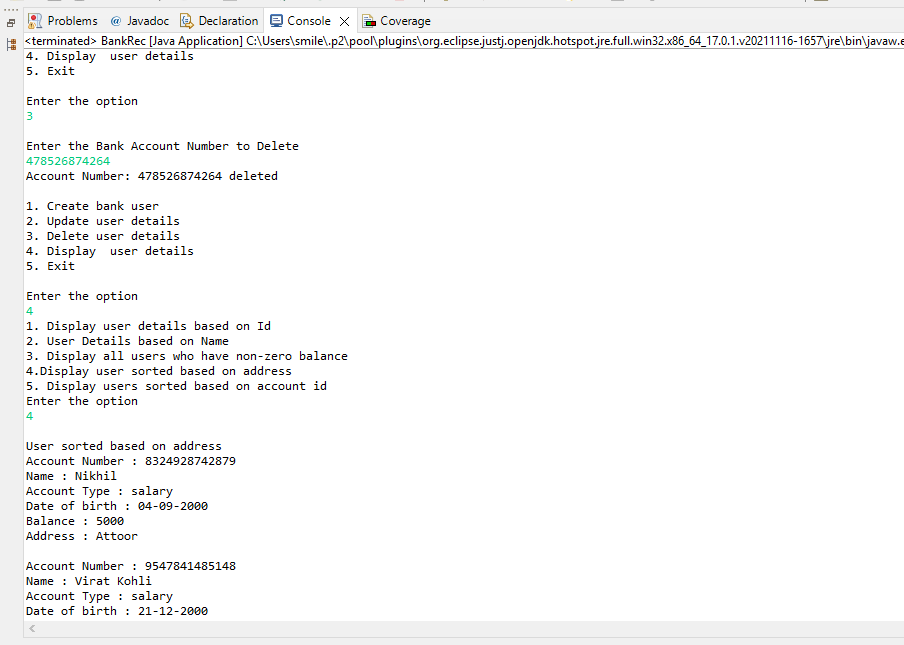
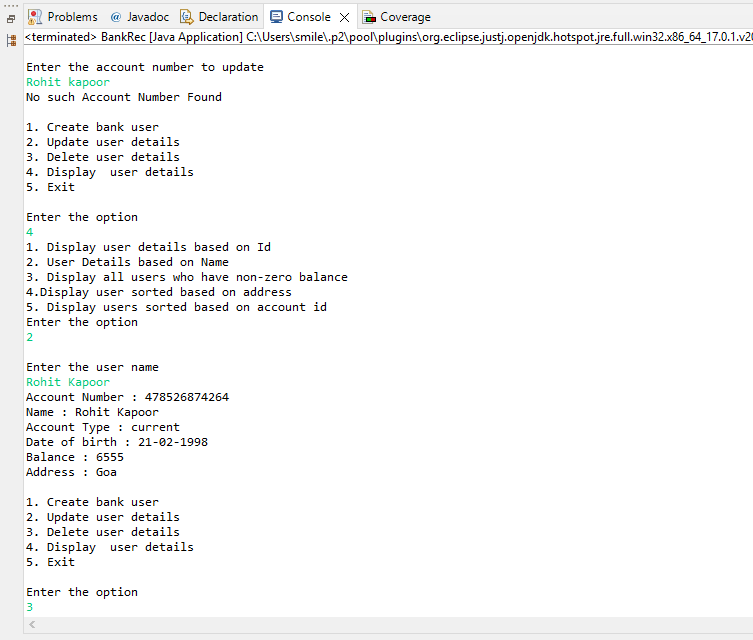
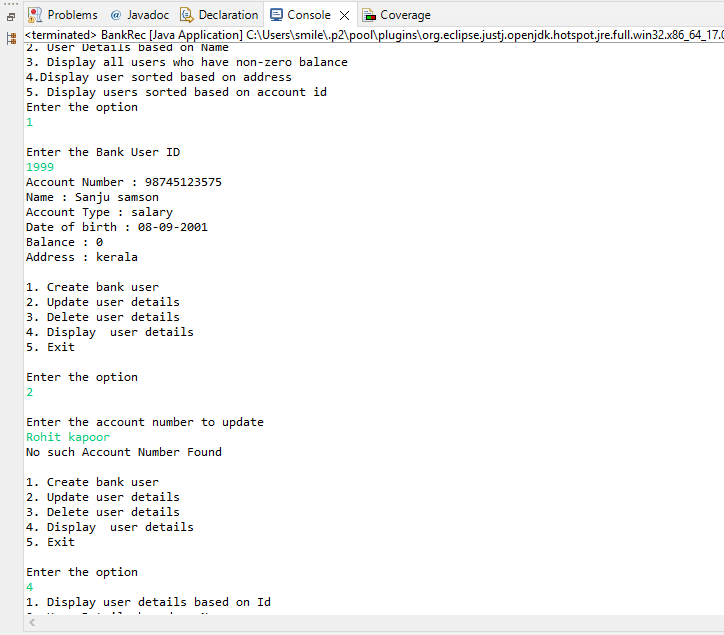
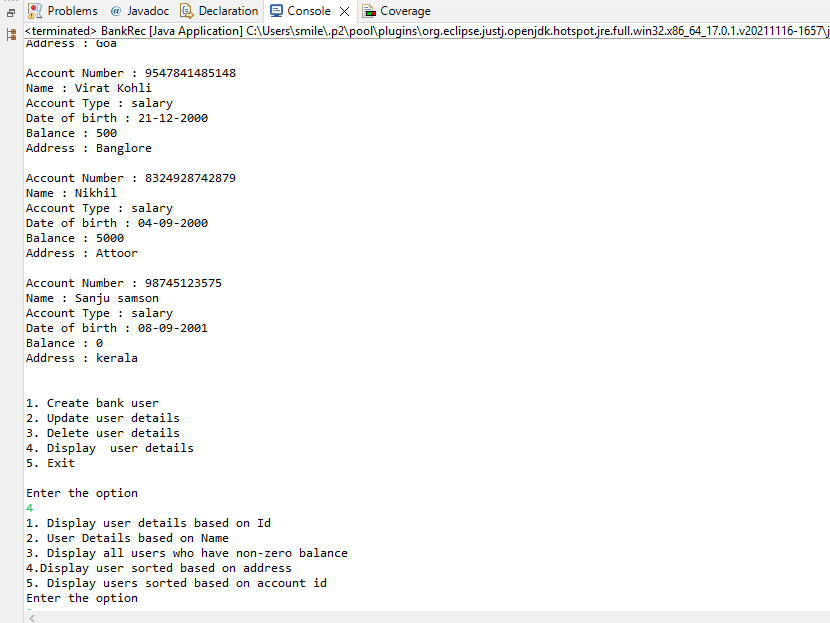
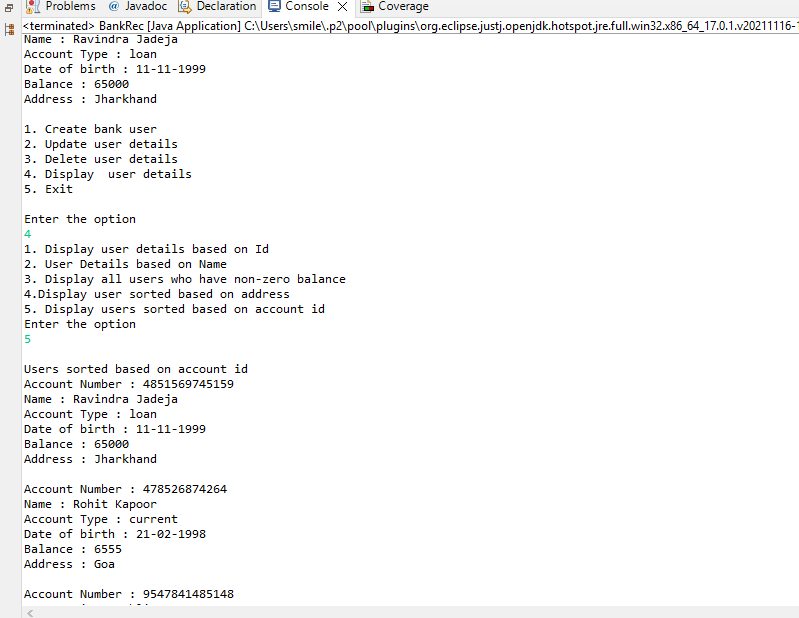
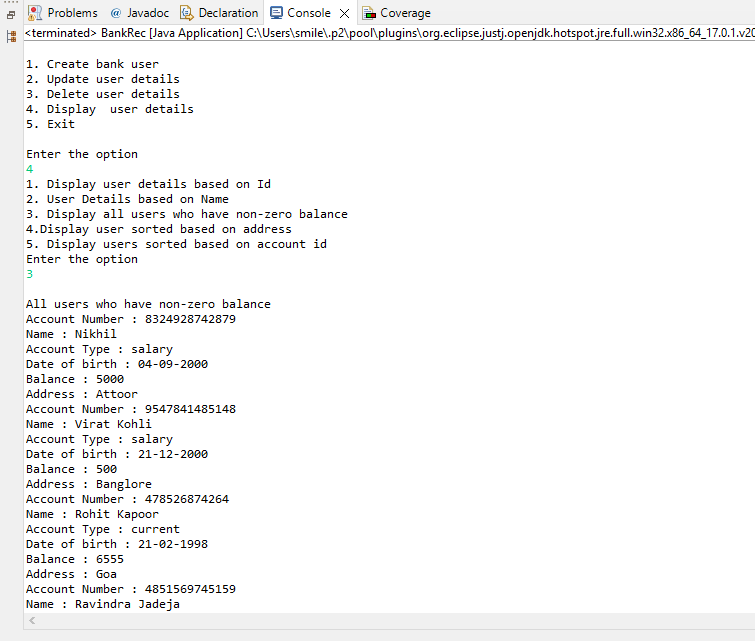
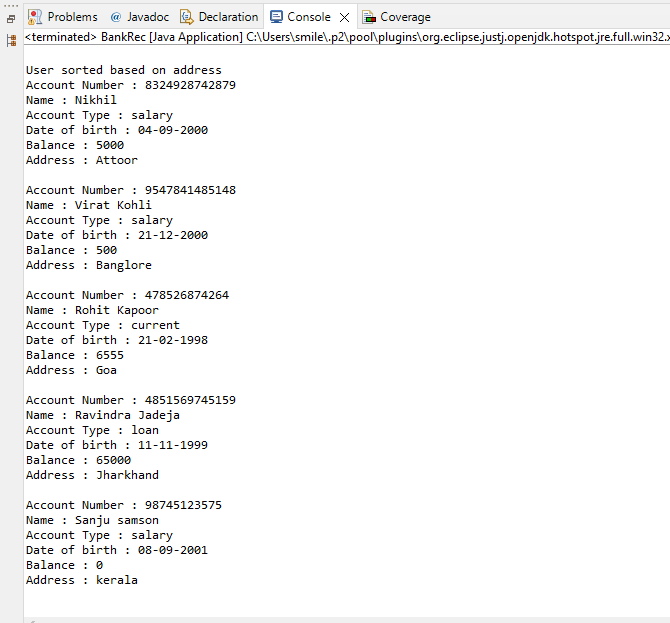
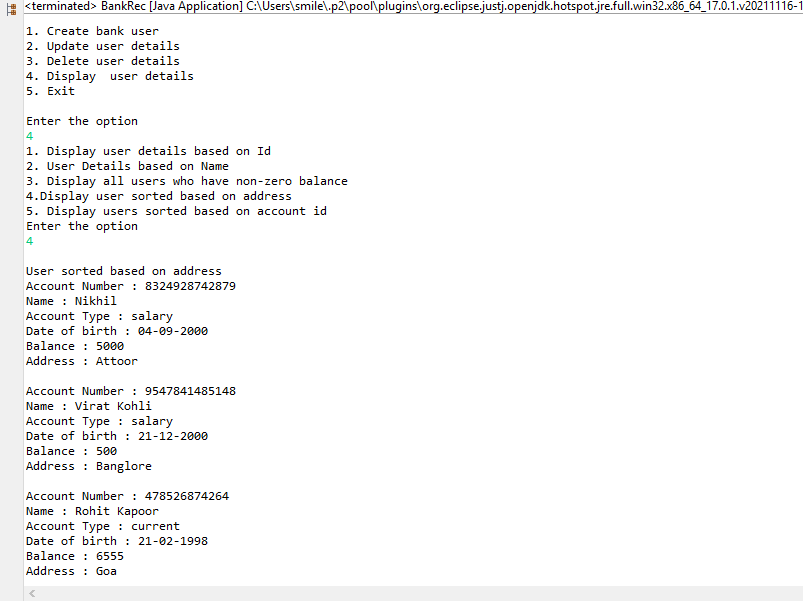
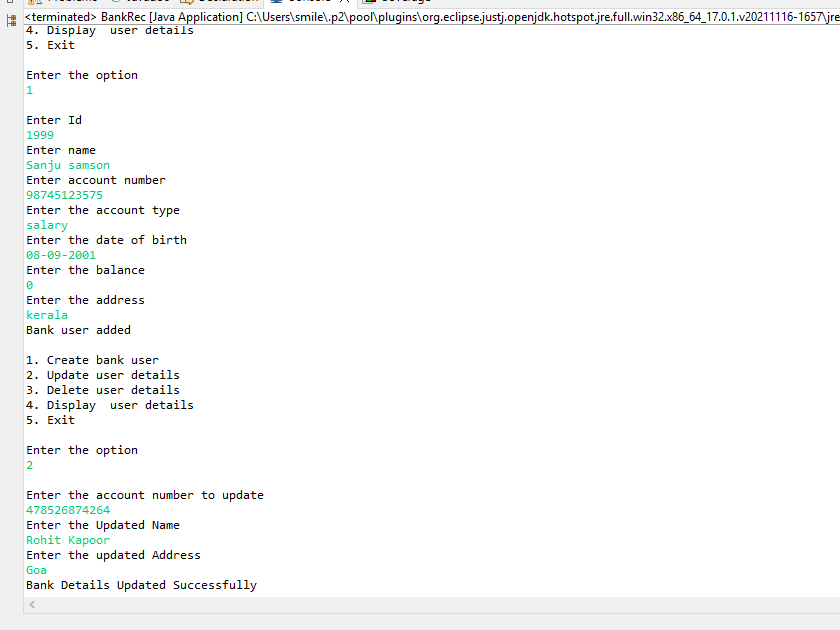
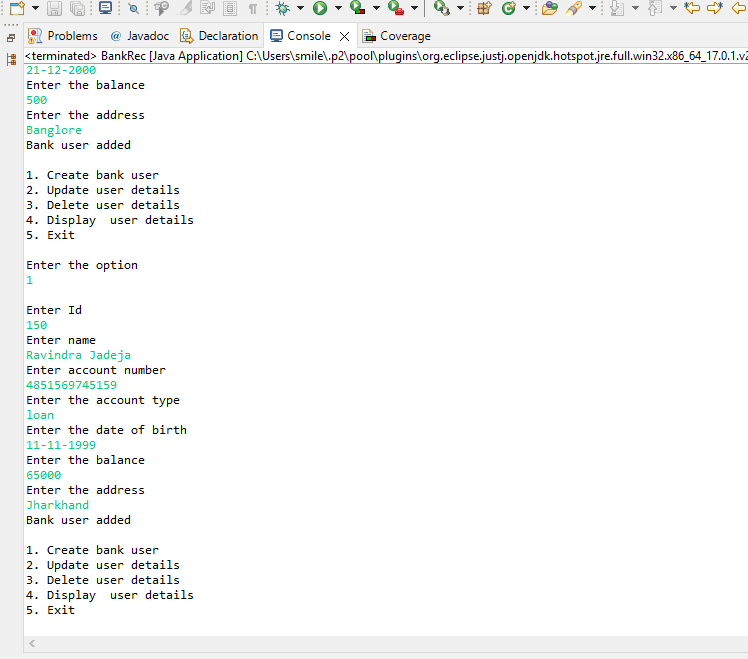
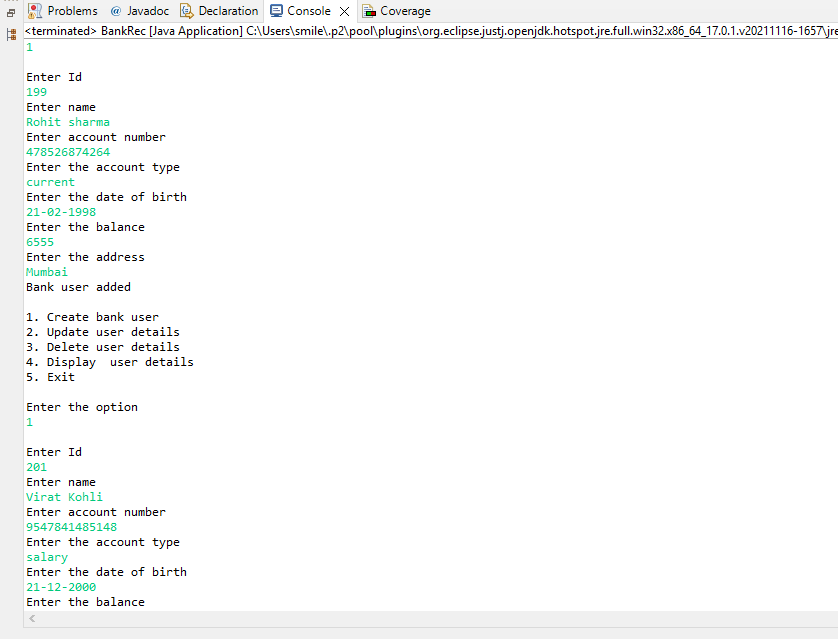
+ ", dateOfBirth=" + dateOfBirth + ", balance=" + balance + ", address=" + address + "]";

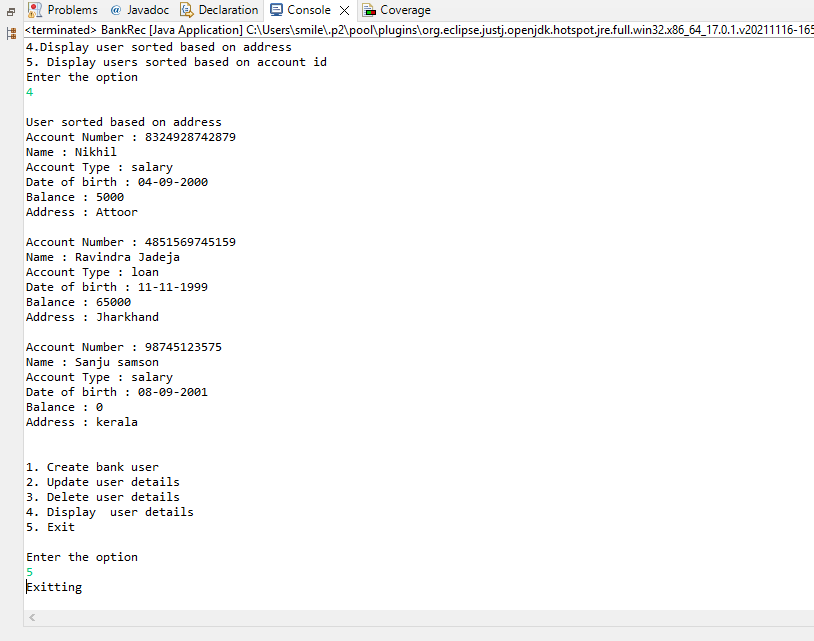
}

}

**OUTPUT:**





****