Assignments

Table of Contents

Introduction to Java

Sr. No.	
1	Write a program to display personal details such as Name, Address, Age Gender, and Qualification.

Variables and Operator

Sr. No.	Assignment Question
1	Write a program for a bank accountant to calculate the interests, which customers will gain on investing any amount of money for any given period of time. (Hint: Keep the rate of interest as fixed.)
2	Write a program to accept two numbers and then display the result of the following operations using the given bit-wise operators: Number 1 & Number 2: Number 1 | Number 2: ~ Number 1: Number 2: (~ Number 1) & (~ Number 2): (~ Number 1) | (~ Number 2): Number 1 >> Number 2: Number 1 << Number 2: Number 2 >> Number 1: Number 2 << Number 1:

Decision-Making and Iterations

Sr. No.	Assignment Question

1	Howell University is an accredited European university, which offers a range of courses to its students. It strives to give students the very best in terms of education and course content.
	
	Now the university management is introducing a new IQ testing system for its MBA students, which is an add-on to the traditional Examination System. This tests the IQ of the students by testing them on four different subjects like Aptitude, English, Mathematics, and General Knowledge.
	
	Consider yourself to be a part of the software development team that is supposed to design the application in Java. First the application asks the student the number of attempt. If student responds with a value higher than 1, then the application terminates and displays a corresponding message.
	
	However, if the student is attempting the test for the first time, it displays the following menu:
	
	Aptitude
	English
	Math
	GK
	Exit
	
	On entering a value between 1 and 4, the application displays the corresponding question. Note that the student can attempt only once for each subject listed.
	
	When a student enters the correct answer for the question, the score for that subject gets incremented by 10 points. Once the student appears the test for all the subjects, he/she can choose to exit the application. After selecting the Exit option, the student can get the total marks printed on the screen.
	
	The total scores should be calculated by adding up all the individual scores in each subject. Next the application displays the following based on the score:
	
	`Bonus points earned`
	`Total score out of 50`
	`Message on IQ level`
	
	The application displays the bonus points based on the following conditions:
	
	1. No bonus point is given, when the total score equals to 10.
	2. A bonus of 2 points is given, when the total score equals to 20.
	3. A bonus of 5 points is given, when the total score equals to 30.
	4. A bonus of 10 points is given, when the total score equals to 40.
	
	The message on IQ level is displayed on the basis of following conditions:
	
	1. If the final score equals 10, then a message is printed saying “Your IQ level is

below average".
2. If the final score equals 22, then a message is printed saying "Your IQ level is average".
3. If the final score equals 35, then a message is printed saying "You are intelligent".
4. If the final score equals 40, then a message is printed saying "You are a genius".
5. If the final score equals 0, then a message is printed saying "You need to re-appear the test".

Introducing Classes

Sr. No.	Assignment Question
1	United Bank is an esteemed bank that provides banking services for profit. Its services include receiving deposits of money, lending money and processing transactions. The management of United Bank is looking at automation as a means to save time and effort required in their work. In order to achieve this, the management has planned to computerize the following transactions: Deposit money in an account Withdraw money from an account The United Bank Manager and a team of experts have chosen your company to provide a solution for the same. Consider yourself to be a part of the team that implements the solution for designing the application. Create an application using Classes and Objects to implement the transactions. The application should consist of the following classes: Account.java AccountTest.java Each file has a specific purpose and functionality. The descriptions of each file are as follows. **Account.java** The Account class is used to store the details of the procedures followed in the bank such as depositing money to the account and withdrawing money from the account. The Account class contains instance variables **amount** and **balance** to store the actual amount and the balance amount respectively. Create a method named **deposit()** to allow the user to deposit some amount in his account. This method should check whether the amount to be deposited in the account is more than zero; otherwise display appropriate message. Similarly, create a method named **withdraw()** to allow the user to amount from his account. This method should check whether the amount to be withdrawn is not more than the current available amount in the account; otherwise display appropriate message. Finally, create a method **getBalance()** to return the balance in the account. **AccountTest.java** The **AccountTest** class demonstrates the use of class, methods and objects in addition to displaying a menu with options of depositing cash, withdrawing cash and quitting the application. An instance of **Account** class is stored in variable **myAccount**.

This class first takes the account number from the user and stores it in the **accountNo** variable. This user-entered number will be validated against the stored value of account number in the **setAccount** variable, if both the numbers are same then the process of depositing and withdrawing money is iterated until the user chooses to exit the application.

Arrays

Sr. No.	Assignment Question
1	The CD House has the most complete collection of Different audio and video CDs, with over 700 individual titles. Now the company wants to introduce automation system to manage the existing and newly added CDs. Consider you to be a part of the team that implements the solution for designing the application. Create an application in Java, which can be used to manage the CD catalog in real time by adding new CDs with all the details captured, searching for existing CDs, displaying the entire catalog with all the details. The application initially displays a menu with the following options: Add CD to the catalog Search CD by CD title Display the catalog Exit Implement the main menu which will display the above list of choices to navigate through the CD catalog. When the user selects an option the corresponding function must be called. The application terminates by printing an appropriate message and showing the corresponding result, when the user chooses to exit the application. When a user enters the first choice, "Add CD", the application checks for space in the array of CDs. If there is place in the array, it starts accepting all the details of the CD one after the other. Otherwise, it prints a message saying "unable to add CD". The details of the CD it accepts are: CD collection name (game/movie/music), CD type (audio or video), Title, Price, CD id and year of release. The details are stored inside the array of CDs and the cdcounter that counts the number of CDs is incremented. For selecting the choice 2 to search a CD by its title, the application checks only if there is CD existing in the catalog. If there is CD in catalog it starts searching by matching the user-entered title with the existing CD titles. If the application finds a match, it returns the details of that particular CD else it prints a failure message. Now, for choosing option 3, which is displaying the CD catalog, it checks the condition, whether there is any CD present in the catalog. The CD details are displayed until all the CDs are displayed. Finally the application exits safely when the option 4 that is Exit is selected.

Packages and Access Specifiers

Sr. No.	Assignment Question
1	RedBox Entertainment is a store which sells music and video CD's. The management is looking at automation as a means to save time and effort required in their work. In order to achieve this, the management has planned to computerize the stock management system. The CEO of the company and a team of experts have chosen your company to provide a solution for the same. Consider yourself to be a part of the team that implements the solution for designing the application. Create an application using different packages and access control specifiers to implement the stock management system. The application should consist of the following entities: 1. A Java package named **music**. 2. A java package named **movies**. 3. A class named CompactDisc in packages, **music** and **movies**. 4. A Java package named **customer**. 5. A class named CDCreator to create and display information about CD's in the package, **customer**. 6. A Java main class named UserInterface inside the **customer** package to display a list of options to the user and run the application. Each file has a specific purpose and functionality. The descriptions of each file are as follows: **CompactDisc.java** The CompactDisc class represents a music album in case it is in the music package and a movie if it is in the movie package. It stores the following details in case of a music album: ➢ title: A String variable to store the title of the album ➢ artist: A String variable to store the name of the artist ➢ price: A double variable to store the retail price of the album ➢ code: An integer variable to store the code for the CD. In case of the CompactDisc class in the movie package the attribute artist will be absent. The instance variables are initialized in the constructor of the class. The CompactDisc class will be used by the CDCreator class to create a music album or a movie CD. **CDCreator.java** The CDCreator class is declared in the package named user. It creates two arrays of objects of the types CompactDisc in the music and movie packages. The CDCreator class uses the following variables for performing various operations on the arrays of CompactDiscs:

- nextMovie: Instance integer variable to store the index of the array where the next movie CD details should be stored.
- nextMovieCode: An integer constant for specifying the maximum number of accounts allowed by the bank.
- nextAlbum: An integer variable to specify the index of the array where the next created album should be stored.
- nextAlbumCode: An integer variable.
- CompactDisc[] cd: An array of objects to store the different music CD details.
- CompactDisc[] vcd: An array of objects to store the different movie CD details.

The object arrays are initialized in the constructor of the CDCreator class. The CDCreator class implements the following methods:

- **void displayAllMusic()**: This method displays the details of the entire music cd array.
- **void displayAllMovies()**:This method displays the details of the entire movie cd array.
- **void addMusicCD()**: This method adds new records to the array of music CD's. If the maximum length of the array is reached then a message is displayed to the user.
- **void addMovieCD()**: This method adds new records to the array of movie CD's. If the maximum length of the array is reached then a message is displayed to the user.

UserInterface.java

This class displays a menu of option to the user to perform the following operations:

- Add a new Music CD to the collection
- Add a new Movie CD to the collection
- Display all music CD's in the collection
- Display all Movie CD's in the collection
- Exit the application

The corresponding methods are invoked when the user chooses an option from the main menu. If the user chooses an incorrect option then a message is displayed to the user. The user for the application will be one of the staff members of the RedBox Entertainment company who will be in charge of handling the items in stock for the company.

Inheritance and Interfaces

Sr. No.	Assignment Question
1	Australian Cricket is still the best team in the world and it has proved its supremacy in the last two World Cups held in the successive years 1999 and 2003. The department handling the finance of the players is planning to develop software that would automatically calculate the income of the players based on their respective grade, the number of matches each player plays and also his performance in the tournament. Also, appropriate income tax should be applied on the income. To accomplish this, a team of experts have been chosen by the Australian Cricket Board to provide a solution for the same. Consider yourself to be a part of the team that implements the solution for designing the application. Create an application using inheritance and interfaces to implement the Software. The application should consist of the following files: 1. Player.java 2. Tax.java 3. PlayerIncome.java 4. GradeBonus.java 5. TournamentIncome.java 6. PlayerTest.java Each file has a specific purpose and functionality. The descriptions of each file are as follows: **Player.java** The Player class is an abstract base class that contains an instance variable to store the name of the Player, and an abstract method. The Booking class should contain an instance variable **name** to store the player's name. An abstract method named **displayDetails()** is created to display the name of the player. **Tax.java** An interface named **Tax** should be created that will act as an interface to calculate the tax of the players income. Declare and initialize a constant variable **TAX_PERCENT** to store the tax percentage. Also create an abstract method named **calculateTax()** to calculate the tax. **PlayerIncome.java** The PlayerIncome class should be a derived class of class Player and implement the interface Tax. This class should be used to store the Player details and display the same.

The PlayerIncome class should contain an instance variable **income** to store the player's income.

Create a new instance of class PlayerIncome and store the salary in the instance variable and call the base constructor.

Now, create a method **calculateTax()** to calculate the tax on respective Player's income.
Override the method **displayDetails()** to display the details about each player.

GradeBonus.java

Again an interface named **GradeBonus** is created to calculate the Bonus given to the player based on his grade.
Declare and initialize a constant variable **GRADE_BONUS_PERCENT** to store the grade bonus percentage. Also create an abstract method named **calculateGradeBonus()** to calculate the bonus to be given to a player on the basis of the grade he possess.

TournamentIncome.java

Again, create a derived class of the Player class and implement the GradeBonus interface. It should be used to store the details of income of each player for each tournament and display the same.

The TournamentIncome class should contain two instance variables **grade** and **rate** to store the player's grade and the rate of each player per match he plays.

Create a new instance of class TournamentIncome and store the details in the three parameters **name**, **grade** and **rate**.

Now, create a method **calculateGradeBonus()** to calculate the bonus of each player based on his grade.

Create a method **displayDetails()** to display the details about each player. Here, calculate the income of the player using his income and the bonus amount he will be given as per his grade.

PlayerTest.java

Finally, create the main class PlayerTest that demonstrates the use of abstract classes and interfaces using the classes PlayerIncome and TournamentIncome. Create an object of PlayerIncome class and an object of TournamentIncome class and pass the argument. Now, display the details of each employee of both the classes using the **displayDetails()** method.

More on Classes

Sr. No.	Assignment Question
1	Alpha Airlines is the premier of the principal U.S. airlines. ATA members transport more than 90 percent of all U.S. airline passenger and cargo traffic. The CEO of Alpha Airlines has now planned to expand the routes upto Singapore and Tokyo. The earlier automated Software could book tickets only for London. But after the addition of two more destinations, that is, Singapore and Tokyo, the existing ticket booking software have to be upgraded. To accomplish this, a team of experts have been chosen by the Airline company to provide a solution for the same. Consider yourself to be a part of the team that implements the solution for designing the application. Create an application using nested class to implement the Software. The application should consist of the following files: ➢ Booking.java ➢ BookingTest.java Each file has a specific purpose and functionality. The descriptions of each file are as follows: **Booking.java** The Booking class represents the actual platform for booking the tickets. The Booking class contains the following variables: ➢ departureDate ➢ departureTime ➢ noOfTickets ➢ price ➢ totalPrice ➢ typeOfCabin Two instance variables named **departureDate** and **departureTime** are initialized in the constructor. This class defines some methods to carry out some operation. The descriptions of each method are as follows: Create a method named **setDepartureDate()** with three parameters for the **day**, **month** and **year** to set instance variable **departureDate**. Create a method **getDepartureDate()** to display the time stored in instance variable **departureDate**. Similarly, create a method named **setDepartureTime()** with two parameters for **hours** and **minutes** to set instance variable **departureTime**. Now, create a method **getDepartureTime()** to display the time stored in instance variable **departureTime**. Also, create a method named **totalPrice()** to compute the total price of tickets

booked. The total price will be decided on the basis of number of tickets, the type of journey and the type of cabin selected by the user.

Since, the concept of nested class is used; three inner classes namely, **Date**, **Time** and **Journey** are created in the Booking class. The descriptions of each class are as follows:

The **Date** class is used to store the date information in three instance variables namely **day**, **month** and **year**. The instance variables are initialized in the constructor of the **Date** class.

Similarly, the **Time** class is used to store the time information in two instance variables namely, **hours** and **minutes**.

Also, another class named **Journey** is used to display the details of the journey. Two instance variables named **destination** and **journeyType** are initialized in the constructor. This class defines few methods to carry out some operation. The descriptions of each method are as follows:

Create a method named **setJourneyDetails()** to accept details, such as entering the destination and the type of journey.

Create a method named **validateJourneyDetails()** to check if the information entered by the user is valid. Since the destination for the Airlines should be only till London, Singapore and Tokyo, a validation should be made for the user to enter only the correct destinations. Another condition should be included in this method to check for the type of journey whether, the user want a single trip or a round trip journey. Based on these conditions, the price for the ticket will be defined.

Create a method named **getJourneyDetails()** to display the details about the journey such as destination and type of journey.

Create a method named **displayTicketInformation()** to display all the information about the tickets booked.

BookingTest.java

Create main class named BookingTest that uses the Booking class to demonstrate nested classes. This program demonstrates the use of Booking class and its nested classes Date, Time and Journey. It accepts and displays the information about airline ticket.

Create an instance of Booking class. Also created an instance of nested classes Date, Time and Journey. Finally, the respective methods should be called to display the complete ticket information in proper format.

Exceptions

Sr. No.	Assignment Question
1	The management of the Merchants Bank is looking at automation as a means to save time and effort required in their work. In order to achieve this, the management has planned to computerize the following transactions: ➢ Creating a new account ➢ Withdrawing money from an account ➢ Depositing money in an account The CEO of the company and a team of experts have chosen your company to provide a solution for the same. Consider yourself to be a part of the team that implements the solution for designing the application. Create an application using exceptions and assertions to implement the transactions. The application should consist of the following classes. ➢ Account.java ➢ Bank.java ➢ BankTest.java ➢ InsufficientFundsException.java Each class has a specific purpose and functionality. The descriptions of each class are as follows. **Account.java** The Account class represents an actual bank account. It stores the following details of a bank account. ➢ customerName ➢ accountNumber ➢ accountBalance The instance variables are initialized in the constructor of the class. The Account class will be used by the Bank class to create bank accounts. **Bank.java** The Bank class creates an array of objects of the Account class to store details of all bank accounts. The Bank class contains the following variables for performing various operations on the bank accounts. ➢ **nextAccount**: Instance integer variable to store the index of the array where the next account details should be stored. ➢ **maximumAccounts**: An integer constant for specifying the maximum number of accounts allowed by the bank. ➢ **nextAccountNumber**: An integer variable to specify the number to be assigned

to the next account created.

- **Account[] account**: An array of objects to store the different account details.

The account array is initialized in the constructor of the Bank class. The Bank class implements the following methods:

- **void displayAccountDetails(Account)**: This method displays the details of the account object passed as an argument to the method.

- **void createAccount()**: This method accepts the user name and the opening balance from the user to create a new instance of the Account class. The method contains an assertion for checking if the opening balance is not negative. Use appropriate try catch blocks to handle all the possible exceptions that can be thrown due to the user inputs. If opening balance is < 100 dollars, then a user-defined exception is thrown.

- **void withdraw()**: This method is used to withdraw money from an account. This method accepts the account number and the amount to be withdrawn from the account. The method then searches in the array of accounts for the account number. Use assertions for checking whether the account number and the amount to be withdrawn are positive. Also use an assertion to check if the array of accounts contains a minimum of one account record. The method also throws the user-defined exception InsufficientFundsException in case the amount to be withdrawn exceeds the account balance. A user must withdraw a minimum amount of 100 dollars for the transaction to proceed. The displayAccountDetails() method is called if the operation succeeds. Use appropriate try catch blocks to handle all the possible exceptions that can be thrown due to the user inputs.

- **void deposit()**: This method is used to deposit money in an account. The account number and the amount to be deposited in the account is accepted from the user. Use an assertion to check whether the account number is positive. The method searches for the account number and deposits the amount in the account if it exists. The displayAccountDetails() method is called if the operation succeeds. Use appropriate try catch blocks to handle all the possible exceptions that can be thrown due to the user inputs. A user-defined exception is thrown if the account number does not exist.

BankTest.java

The BankTest class is a java main class used to test the Bank class. It creates an instance of the Bank class and displays the following menu of options to the user.

- Create a new account
- Withdraw Cash
- Deposit cash
- Exit

The user can select any of the options and a corresponding method is invoked on the instance of the Bank class. Use an assertion to check for the control-flow invariant in case the user types an invalid option. The application exits when the Exit option is

selected.

InsufficientFundsException.java

This is a user-defined exception class derived from the base class Exception. This exception is thrown when the user tries to withdraw more money than the current account balance. The InsufficientFundsException class has two constructors. One is the default constructor which calls the constructor of the super class. The second constructor accepts the account balance as an argument. The current account balance is displayed in the message when the exception is thrown.

--- End of Assignments ---