Revision Core Java – Exception Handling & Inheritance

Q1. XYZ software Company is working on a Bank Project which follows the below given design. You need to implement this project as per the guidelines mentioned below:

Design an interface named as Role which provides the following two contracts:

```
public String getRoleName();

public String getResponsibility();
```

Design two classes CEO and Manager as two Roles that implement the Role interface mentioned above and provide the definition of two contracts mentioned in the Role interface.

Design a main class which instantiate the two object one which is a CEO object another is Manager object, display the Role Name and Responsibility of two objects accordingly.

Q2 Create a class named “CollegeCourse”, that includes the following data fields

- department (for example,“ENG”),
- course number (for example, 101),
- credits (for example, 3),
- fee for the course (for example, $360).

All of the fields are required as arguments to the constructor, except for the fee, which is calculated at $120 per credit hour. Include a display () method that displays the course data.

Create a subclass named “LabCourse” that adds $50 to the course fee. Override the parent class display() method to indicate that the course is a lab course and to display all the data.

Write an application named UseCourse that prompts the user for course information.

If the user enters a class in any of the following departments, create a LabCourse: BIO, CHM, CIS, or PHY.

If the user enters any other department, create a CollegeCourse that does not include the lab fee. Then display the course data.

Save the files as CollegeCourse.java, LabCourse.java, and UseCourse.java.

Q3)

Write an application named UseChildren that creates and displays at least two Child objects—

one Male and one Female.

Child is an abstract class and Male and Female are subclasses.

The Child class contains fields that hold the name, gender, and age of a child.

The Child class constructor requires a name and a gender.

The Child class also contains two abstract methods named setAge()and display().

The Male and Female subclass constructors require only a name;

they pass the name and appropriate gender to the Child.

The subclass constructors also prompt the user for an age using the setAge() method, and display the Child's data using the display()method.

Save the files as Child.java, Male.java, Female.java, and UseChildren.java.

Q4)

Write an application named GoTooFar in which you declare an array of five integers and store five values in the array.

Write a try block in which you loop to display each successive element of the array, increasing a subscript by 1 on each pass through the loop.

Create a catch block that catches the exception ArrayIndexOutOfBoundsException and displays the message, "Now you've gone too far"

Q5)

The Integer.parseInt() method requires a String argument, but fails if the String cannot be converted to an integer.

Write an application in which you try to parse a String that does not represent an integer value.

Catch the NumberFormatException that is thrown, and then display an appropriate error message.

Q6)

Write an application that prompts the user to enter a number to use as an array size, and then attempt to declare an array using the entered size.

If the array is created successfully, display an appropriate message. Java generates a NegativeArraySizeException if you attempt to create an array with a negative size, and a NumberFormatException if you attempt to create an array using a nonnumeric value for the size.

Use a catch block that executes if the array size is nonnumeric or negative, displaying a message that indicates the array was not created.

Q7)

Write an application that throws and catches an ArithmeticException when you attempt to take the square root of a negative value. Prompt the user for an input value and try the Math.sqrt()method on it. The application either displays the square root or catches the thrown Exception and displays an appropriate message.

Q8)

Create an EmployeeException class whose constructor receives a String that consists of an employee's ID and pay rate.

Save the file as EmployeeException.java. Create an Employee class with two fields:

idNum and hourlyWage.

The Employee constructor requires values for both fields.

Upon construction, throw an EmployeeException if the hourlyWage is less than $6.00 or over $50.00. Save the class as Employee.java.

Write an application that establishes at least three Employees with hourlyWages that are above, below, and within the allowed range.

Display an appropriate message when an Employee is successfully created and when one is not.

Q9)

Write an application that displays a series of at least five student ID numbers (that you have stored in an array) and asks the user to enter a numeric test score for the student.

Create a ScoreException class, and throw a ScoreException for the class if the user does not enter a valid score (less than or equal to 100). Catch the ScoreException and then display an appropriate message.

In addition, store at 0 for the student's score. At the end of the application, display all the student IDs and scores.

Q10) Explain the following topics with scenario based examples

1. Explain the Exception class hierarchy
2. Difference between checked and unchecked exceptions.
3. The use of try & catch
4. The relevance of throws and throw
5. The relevance of finally
6. How to create a custom Exception
7. Abstract class & interface
8. Object class in java