Java Assessment Questions

1. write a program to calculate Area of -

* Rectangle
* Circle
* Triangle

Make use of Appropriate annotations and javadoc .

1. Design a Java program that simulates an ATM machine. Prompt the user to enter their account balance and the amount they want to withdraw. Handle the scenario where the user enters non-numeric or negative values for balance or withdrawal amount. If any invalid inputs are detected, throw a custom exception called **InvalidInputException** with an appropriate error message. Implement the **InvalidInputException** class.
2. Write a Java program that contains a method to calculate the area of a rectangle. The method takes two positive integer parameters representing the length and width of the rectangle. However, the method should throw a custom exception called “InvalidDimensionException” if either the length or width is less than or equal to zero. Implement the “InvalidDimensionException” class with an appropriate message. Test the method with valid and invalid inputs.
3. Create a Java program that reads data from a text file specified by the user. The program should handle potential exceptions such as the file not found, permission issues, or other file-related exceptions gracefully using proper exception handling techniques. Provide appropriate error messages for each type of exception.
4. Write a Java program that copies the content of one text file to another. The program should take the filenames of the source and destination files as user input. Implement exception handling to deal with potential **IOExceptions** that may occur during file reading and writing. Provide appropriate error messages for file-related exceptions.
5. Design a Java program that defines a list of strings. The program should prompt the user for an index and display the element at that index. Handle the scenario where the user enters an invalid index (out of bounds) or tries to access an element from a null list. Throw appropriate exceptions (**IndexOutOfBoundsException** and **NullPointerException**) with informative messages.
6. Write a Java program that initializes an array of integers and prompts the user for an index to access an element. Handle the scenario where the user enters an invalid index (out of bounds) or tries to access an element from a null array. Throw appropriate exceptions (**IndexOutOfBoundsException** and **NullPointerException**) with informative messages.
7. Write a Java method that reads data from a text file specified by the user. The method should take the filename as a parameter and use the **finally** block to ensure that any resources (e.g., file streams) are closed properly, regardless of whether an **IOException** occurred during file reading or not.
8. Create a Java program that prompts the user for an integer input and checks if it is an even number. If the number is even, print "It's an even number." If the number is odd or non-numeric, throw a custom exception called **NotEvenNumberException** with an informative message. Implement the **NotEvenNumberException** class.
9. Design a Java program that asks the user to enter a password. Check if the password meets the criteria of having at least eight characters, containing both letters and numbers. If the password does not meet the criteria, throw a custom exception called **InvalidPasswordException** with a suitable message. Implement the **InvalidPasswordException** class.