Submission

- Please include these files in your submission:

1. `.pdf` of code and demo result
2. `.zip` of the `src/` folder

Coding

Write code in Java to solve following problems. Please write your own answers. You are highly encouraged to present more than one way to answer the questions. Please follow best practice when you write the code so that it would be easily readable, maintainable, and efficient. Clearly state your assumptions if you have any. You may discuss with others on the questions, but please write your own code.

1. Write a Java Program to create an abstract class named `Shape` that contains two integers and an empty method named `printArea()`. Provide three classes named `Rectangle`, `Triangle` and `Circle` subclass that each one of the classes `extends` the Class `Shape`. Each one of the classes contains only the method `printArea()` that prints the area of Shape.
2. Guess Game Project
    1. **Develop a simple class that represents a number guessing game.**
        - The game is played by the program randomly generating a number and the user attempting to guess that number. After each guess the program will provide a hint to the user identifying the relationship between the number and the guess. If the guess is above the answer then “Too High” is returned, if the guess is below the answer then “Too Low”.
        - Also if the difference between the answer and the guess is less than the difference between the answer and the previous guess, “Getting warmer” is returned. If the difference between the answer and the guess is more than the difference between the answer and the previous guess, then “Getting Colder” is returned.
    2. **The program will allow the user to play multiple games.** Once a game is complete the user will be prompted to play a new game or quit.
    3. **Design and build a GuessingGame class.**
        1. Seven instance variables.
            1. answer - an integer representing the randomly generated number.
            2. generator – a random Generator object
            3. gameOver – a Boolean, false if game still in progress, true if the game is over.
            4. differential – an integer representing the difference between a guess and the answer.
            5. max – maximum value of the number to guess. For example, if the maximum number is 100 then the number to guess would be between 0 and 100.(inclusive)
            6. maxGuessesAllowed – the maximum number of guesses the user gets, once this value is passed the game is over.

7. numGuessesTaken – an integer that stores the number of guessed taken so far in any game.

2. Constructor and Methods
    1. Default Constructor Sets max to zero Generates the random number generator object.
    2. Parameterized Constructor Takes an integer parameter representing the maximum value of the number to guess. Creates the random number generator object.
    3. newGame method Takes in an integer as a parameter representing the maximum number of guesses and sets maxGuessesAllowed. In other words the parameter represents how many guesses the user gets before the game is over. Generates the answer using the random number generator.(0 - max). Sets gameOver to false. Sets differential to the max value. Sets numGuessTaken to zero.
    4. guess method Takes an integer as a parameter representing a new guess. Compares the new guess with the answer and generates and returns a String representing an appropriate response. The response is based on:
        1. The relation of the guess and answer (too high, too low or correct).
        2. The comparison of difference between the current guess and the answer and the previous guess and the answer. (warmer, colder)
        3. Guess out of range error, if the guess is not between 0 and the max number (inclusive) (see sample run below)
        4. User has taken too many guess because numGuessesTaken is greater than maxGuessesAllowed. If this is the case set isGameOver to true.
    5. isGameOver method - returns the state of game.
        1. true if game is over
        2. false if still in progress.
    6. Accessor(getter) and mutator(setter) methods for all instance fields except the Random number generator. Use the Accessor or mutator methods within the other methods of the class rather than directly accessing the instance fields. For example, use mutator methods in the parameterized constructor to modify instance variables.

4. **Design and build GuessingGameTester class**
    1. This program will create GuessingGame objects and completely test the GuessingGame Class.
    2. The tester will also provide two loops.
    3. The first loop will allow the user to play a new game after the previous game is completed.
    4. The second or nested loop will prompt the user for a new guess and provide a response based on the guess.

Coding Sample

Welcome to the Guessing Game
Enter the maximum number
100
answer is: 8 (Included for testing only, should not be display in final program)
Enter the number of guess allowed:
6
Enter your guess, remember it must be between 0 and 100
50
Too High
Getting Warmer

Enter your guess, remember it must be between 0 and 100
25
Too High
Getting Warmer

Enter your guess, remember it must be between 0 and 100
12
Too High
Getting Warmer

Enter your guess, remember it must be between 0 and 100
6
Too low
Getting Warmer

Enter your guess, remember it must be between 0 and 100
8
Congratulation

Would you like to play again, enter Y for yes, N for no.
Y
Welcome to the Guessing Game
Enter the maximum number
50
answer is: 36

Enter the number of guess allowed:
5
Enter your guess, remember it must be between 0 and 50
60
Guess out of range, The guess must be between 0 and 50
Enter your guess, remember it must be between 0 and 50
25
Too low
Getting Warmer

Enter your guess, remember it must be between 0 and 50
48
Too High
Getting Colder

Enter your guess, remember it must be between 0 and 50
37
Too High
Getting Warmer

Enter your guess, remember it must be between 0 and 50
36
Congratulation
Would you like to play again, enter Y for yes, N for no.
N