Question 1: Guess the number! [15 Marks]

In the main method of the class **GuessNumber** of the skeleton code provided to you, implement a program generating a single random number and then requiring the user to guess it. A method is provided to you, **int posRandNum(int max)**, to randomly generate an integer in the range from **1** to **max**.

Question 1.1 [12 marks]
Implement code to guess a random number in the range [1..10], following the guidelines below. A typical interaction with the user of the program is shown below, detailing:

- The output for a few incorrect guesses of the random number;
- The output for a correct guess of the random number.

Note: *Assume the **nextInt()** method in the **Scanner** class is sufficient to scan user input. Your program must work for any valid input and you can assume the input is always valid.*

```
Guess the random number!
1
Incorrect guess!
---------------------------
Guess the random number!
5
Incorrect guess!
---------------------------
Guess the random number!
4
Incorrect guess!
---------------------------
Guess the random number!
2
Congratulations, you guessed the right number!
```

Question 1.2 [3 marks]
Modify the code to detect whether the number being incorrectly guessed by the user is within **2 units** from the random number. For instance, if the random number being guessed is 6 then 7, 8, 5 and 4 are all within 2 units. A revised interaction with the user is provided below, showing:

- The output for an incorrect guess of the random number within 2 units;
- The output for an incorrect guess of the random number;
- The output for a correct guess of the random number.

```
Guess the random number!
7
You're pretty close to the mark now!
---------------------------
Guess the random number!
9
Incorrect guess!
---------------------------
Guess the random number!
6
Congratulations, you guessed the right number!
```

Question 2: GameCharGen - Main Menu [15 Marks]

A local game developer, **The League of Über Nerds**, has commissioned you the prototype of a random character generator for their latest computer role-playing game, **World of Kitschcraft**. This early prototype will only model a subset of attributes for the game characters, namely:

- Name (The game character's name)
- Hit points (The game character's vitality/energy level)
- Strength (The game character's strength)
- Intelligence (The game character's intelligence)
- Wisdom (The game character's wisdom)

Moreover, there are exactly three game character types, namely:

- Fighter
- Wizard
- Cleric

Those character types are implemented in a **Fighter**, **Wizard** and **Cleric** class, derived from a **GameCharacter** base class. Each derived class overrides two methods:

- A **void randGen(boolean areHitPointsMaxed)** method. This method randomly generates the values of the game character's attributes. However, when the boolean parameter maxHitPoints is set to **true**, it assigns a predetermined maximum value of hitpoints for any game character type.
- A **void display()** method. This method outputs a game character's attributes similarly to the output below:

```
Name:           Felonious
Class:          Wizard
HitPoints:      6
Strength:       12
Intelligence:   17
Wisdom:         16
```

Question 2.1 [15 marks]
Implement a console-based menu in the main method of a class named **GameCharGen**, included in the skeleton code provided to you. The menu will carry out the commands illustrated below.

```
*** WoK GameCharGen Menu***
Create Wizard           WZ
Create Wizard (MAX)     WZM
Create Fighter          FG
Create Fighter (MAX)    FGM
Create Cleric           CL
Create Cleric (MAX)     CLM
Exit Program            EX
Enter selection:
```

You must implement the menu as shown. Do not change the options or the inputs for selecting the menu options. Every time you activate the creation of a game character, you should firstly call the **randGen()** method to randomly generate it (the max options call **randGen()** with maxHitPoints set to **true**, while the others set maxHitPoints to **false**). Subsequently, you should invoke the **display()** method to print out the game character attributes.
Note: *The letters on the right represent what the user must type to use that feature. For instance, to create a new Wizard, the user must input 'WZ'. The solution should be case insensitive therefore the user should be able to also enter either 'wz', 'Wz' or 'wZ', etc.*

Question 3: GameCharGen - Find Game Characters [20 Marks]

Building from Question 2, every time a game character is randomly generated, it gets stored in an array named **randomGameChars** in the class **GameCharGenModel**, included in the skeleton code provided to you. Based on that assumption, implement the additional functionalities specified below.

Question 3.1 [10 marks]
Implement the **int findGameCharacter(GameCharCapabilities.Attribute attribute)** method provided, which returns the index in the array of either the wisest, the strongest or the most intelligent game character (based on a parameter of type **Attribute**, see the **GameCharCapabilities** interface). If the game character cannot be found, then it throws a custom exception **NoGameCharacterFoundException**. This exception class has already been implemented in the skeleton code provided to you. Also, if more than one game character has maximum wisdom, strength or intelligence the method only returns the first.

Question 3.2 [5 marks]
Implement the **int findGameCharacterNearestToAverage()** method provided that finds the game character closest to the "average game character" in the array and returns its index. If the method cannot find the average game character, then it throws a custom exception **NoGameCharacterFoundException**. This exception class has already been implemented in the skeleton code provided to you. The average game character is calculated by finding the average attribute values for intelligence, strength and wisdom across all generated characters. After that, the distance of each game character from the average one is calculated as the difference from the average attributes. This allows the determination of which game character is closest to the average. Pseudo-code for the calculation is given below.

```
IF no game characters found TERMINATE
  COMPUTE AVERAGE_LEVEL for each attribute from all game characters
    FOR EACH game character
        FOR EACH attribute COMPUTE DISTANCE as difference from the attributes average.
        COMPUTE TOTAL_DISTANCE as sum of each DISTANCE
        DETERMINE if game character is closer than the previous game character
  RETURN location of closest game character to the average levels
```

Question 3.3 [5 marks]
Add four menu options to the class **GameCharGen** as below. The options will display the strongest, wisest and most intelligent game characters generated so far, using the method implemented in **3.1**, and the "average" game character using the method implemented in **3.2**.

```
*** WoK GameCharGen Menu***
Create Wizard               WZ
Create Wizard (MAX)         WZM
Create Fighter              FG
Create Fighter (MAX)        FGM
Create Cleric               CL
Create Cleric (MAX)         CLM
Display Most Intelligent    DI
Display Wisest              DW
Display Strongest           DS
Display Average             DA
Exit Program                EX
Enter selection:
```

Note: *The letters on the right represent what the user must type to use that feature. Once again, the solution should be case insensitive:the user should be able to also enter either 'wz', 'Wz' or 'wZ', etc.*

End of Exam