**Weeks 5-6 Lab, Part 2**

Do Chapter 5 Exercise **5** (an *acceptable integer* class) and Practice Programs **3** (a *Magic8Ball* class) and also do **1** (comparing future *Species* populations) **for extra credit.**

Do <u>Chapter 5</u> Exercise **5**, creating an ***acceptable integer value*** class:

- Consider a Java class that you could use to get an <u>acceptable</u> **integer value** from the user. An object of this class will have these attributes:

  – the minimum accepted *int* value (inclusive)

  – the maximum accepted *int* value (inclusive)

  – a prompt String for asking for the integer

  and the following method:

  – ***getValue()*** displays/prints the prompt and reads a value using the class *Scanner*. If the value read is <u>not</u> within the allowed range, the method should display an error message and ask the user for a new value, <u>repeating</u> these actions in a validation *do-while* loop until an acceptable value is entered. The method then <u>returns</u> the *int* value read.

- **Write <u>precondition</u> and <u>postcondition</u> <u>comments</u> for the method *getValue()*.**

- Implement a ***ValidInt*** class in Java. You will need to create an ***initialize()*** method that sets all three instance variables, create and initialize some ***ValidInt*** objects in a *main* method, and write Java statements to <u>test</u> ***getValue()*** in *main*.

1. Do <u>Chapter 5</u> Practice Program **3**, creating a ***Magic8Ball*** <u>class</u>:

   - Redo or do Practice Program 8 from Chapter 4, except write a **class** for the Magic 8 Ball. The class should have a **method** that returns an answer at random by generating a <u>random number</u> that allows choosing one of the following 8 responses; you can call that method ***getAnswer()*** – it should take <u>no parameters</u> and <u>return</u> a ***String***.

   - These Java statements will generate a random number between 1 and 8:

     ```
     import java.util.Random; // or import java.util.Random;
     static Random r = new Random(); // put both outside/above all methods
     int num = r.nextInt(8) + 1; // put this inside the getAnswer() method
     // use num in an if or switch statement to return a response below
     ```

   - These are the 8 responses that ***getAnswer()*** should randomly choose from and <u>return</u>:

     | | | |
     |---|---|---|
     | **1. It is certain** | **2. It is decidedly so** | **3. Most likely** |
     | **4. Signs point to yes** | **5. Reply hazy, try again** | **6. Ask again later** |
     | **7. Don't count on it** | **8. My sources say no** | |

   - Test your class by creating a *Magic8Ball* object and printing what its ***getAnswer()*** method returns several times – it should print several different answers, but they are <u>not</u> guaranteed to all be different.

2. Do Chapter 5 Practice Program **1**, comparing future **Species** populations (**extra credit exercise**):

- Write a program to help answer questions like the following: Suppose the species **Klingon ox** has a population of **100** and a growth rate of **15** percent, and the species **elephant** has a population of **10** and a growth rate of **35** percent. How many years will it take for the elephant population to exceed the Klingon ox population?

- Use the version of the class **Species** from Sakai's Week 6 **Source Code** folder as your starting point – download it to your **Java programs** folder. Your program will ask for the data on both species and will respond by telling you how many years it will take for the species that starts with the lower population to *outnumber* the species that starts with the higher population. The two species may be entered in any order, so you'll have to figure out which is which; you might assign them to **lower** and **upper** *Species* variables. **You can assume the populations are different to start.**

- In a *main* method in this program or a separate program, create two *Species* objects and run a loop with increasing future year numbers until the lower population species outnumbers the higher population species, using the **predictPopulation()** method. Note that it's possible the species with the smaller population will never outnumber the other species. In this case, your program should just display a suitable message.

- *Hint*: **If the species with the lower population has a growth rate less than or equal to the species with the higher population then it will never outnumber the other species, otherwise it must outnumber it at some year in the future. You may assume that both species have growth rates greater than 0 for this exercise.**

Show me how you have completed these exercises.