Lab 16 – Lucky Dice

Open VisualStudios, and create a new project titled **Lab16-LuckyDice** in your CS\LABS folder.

Create a new class and **type** in our code skeleton:

```
//Name:
import java.util.*;
public class PracticeProblems
{
	public static void main(String[] args)
	{
		Scanner console = new Scanner(System.in);
	}
}
```

Before each problem, insert a COMMENT with the problem number.

1. Get an integer from the keyboard, and print all the factors of that number. Example, using the number 24:

   ```
   Factors of 24 >>> 1 2 3 4 6 8 12 24
   ```

2. Practice using **arrays** by adding all the factors into an **int[]** array.

3. Print first and last integers by calling the array

   ```
   First and last numbers >>> 1 24
   ```

4. Write a new version of problem 1 that gets a random number between 10 and 20 instead of asking the user to provide the number.

5. Replace values in the **int[]** array using the new factors

6. Repeat step 3

7. Write a **for** loop that prints (on a single line) 15 random numbers between -5 and 5, inclusive. (Inclusive means that both -5 and 5 are possible numbers that might get chosen.)

8. Write a **while** loop that randomly chooses (and prints in a line numbers between 31 and 33, inclusive. The loop should only stop when all numbers within the range have been selected by your random number generator. This can easily be accomplished using 3 boolean flags. Then print `"Got them all!"`

   ```
   Can I get them all? >>> 31 33 31 31 33 32
   Got them all!
   ```

9. A "perfect number" is a number that equals the sum of its divisors (not including the number itself). Example: 6 is a perfect number. (The divisors of 6 are 1, 2, and 3. Adding those divisors (1 + 2 + 3) put you back at 6!)

   Get an integer from the keyboard and write the code to determine if it's a perfect number by using a **for** loop.

10. Make a rock/paper/scissors game where the user plays against the computer. Use a **do while** loop to loop the game until the human selects option 4. (See sample output)

```
Hint 1) information on do while loops is available in the while loop notes (Lesson 11)

Hint 2) Consider that a game of RPS only has 9 possible outcomes, so using an if statement
for each scenario is a reasonable approach.
```

Lucky dice app

Create a new class and **type** in our code skeleton:

```
//Name:
import java.util.*;
public class LuckyDice
{
	public static void main(String[] args)
	{
		Scanner console = new Scanner(System.in);
	}
}
```

One of the most essential uses of a computer (aside from YouTube, apparently) is to **simulate** what might happen in the real world, so that we can make informed decisions in all kinds of complicated situations that wouldn't be possible or feasible to actually test ourselves.

Write a program that will calculate the average number of **rounds** it takes to reach 1000 total 'points'. In each **round,** 2 dice are rolled and their values are added to the total. (Even though 2 dice are rolled, that still counts as only 1 round.)

Your program should ask the user for the number of faces on the die (you can't assume it will be an ordinary 6-sided die) and the number of simulations they'd like to run. An example run of the program is shown below (**user input shown in red**):

```
Number of faces on the die? >>> 20
Number of simulations? >>> 5

Simulation 1: number of rounds to reach 1000 >>> 51
Simulation 2: number of rounds to reach 1000 >>> 53
Simulation 3: number of rounds to reach 1000 >>> 41
Simulation 4: number of rounds to reach 1000 >>> 47
Simulation 5: number of rounds to reach 1000 >>> 41
Average number of rounds for all simulations >>> 46
```

You should use a for loop to run the number of simulations, and a while loop (inside the for loop) to control each simulation. The inner while loop will run while the current number of 'points' is less than 1000.

You will need variables to store this information! **Each simulation** starts with the number of rounds and the amount of points at 0. You'll also need to store the **total** amount of rounds it takes (for every simulation) so you can calculate an average. Pseudo code:

```
for the number of simulations entered
```

```
initialize number of rounds and points to 0
while current points are less than 1000
        roll both dice (random number based on number of faces)
        increment round counter
calculate and print the average
```

(Advanced/Optional) Gambler's ruin

Suppose a gambler makes a series of $1 bets, starting with $50 of cash, and continues to play a casino game until she either goes broke or has won $250. What are the chances that she will go home with $250, and how many bets would it take before she won or lost?

Write a program that can help answer these questions. It reads in values for four variables (from the keyboard) prior to running the simulation:

- the initial stake (amount of money you start with)
- the goal amount (how much you'd like to walk away with, your winnings)
- the probability of winning whatever game she's playing* (as an integer)
- the number of simulations to run

**for example, the probability of winning a hand in Blackjack is roughly 43%*

This program will require two loops, one inside of the other. The outer loop (most likely a for loop as we know how many times it will run) will run the number of simulations requested, and the inner loop (a while loop might be useful, as we don't know how long it will run beforehand) runs the 'current' simulation for the gambler, and continues to run until they either go broke or win the desired amount (a loop inside a loop is called a nested loop).

To determine if the gambler won or lost, your program should generate a random number from 1 to 100. If the number generated is below the probability of winning, the gambler "wins the hand" and $1 is added to their winnings. Otherwise, $1 is subtracted.

The following code will generate a random number and save it into variable *a*:

```
Random randomGen = new Random();
int a = randomGen.nextInt(100) + 1; //generates new random int from 1 to 100
```

Click here to watch a video of a sample run of the program.

Remember, if you declare a variable inside a loop, it will continue to be re-initialized every time through the loop!