# C++ Loaded Dice OOP

## Contents

Time required: 120 minutes

- Comment each line of code as shown in the tutorials and other code examples.
- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

---

## Pseudocode

1. Write pseudocode or TODO for the exercise

## Part 1: Random Dice

Dice are used in many games. One die can be thrown to randomly show a value from 1 through 6.

Create a C++ file named **Die.cpp**

The **Die** class rolls and returns one random int dice value. There isn't any other functionality in the **Die** class.

1. Design a C++ **Die** class with a data field for an integer value.
2. Create a default constructor with no parameters.
3. Create the proper header (.h) and implementation file (.cpp).

4. Create a **rollDie()** method. This method generates a random number between 1 and 6 and assigns it to a member variable.

5. The **rollDie()** method returns the value generated.

This example assumes you have assigned appropriate values to the static constants.

```
#include <ctime>   // For time function
#include <cstdlib> // for rand srand functions
// rand gives the same sequence each time the program runs
// Initialize random number generator with different values
// time(0) Time since January 1st, 1070 at 00:00:00 AM
srand(time(0));

int randomDie = rand() % MAX_VALUE + MIN_VALUE;
```

## Part 2: Main Program

Build and test the main program one step at a time.

1. Create two die objects.

2. Roll and display a single die roll from each die object.

3. Compare and display the roll (returned int) from each die object to determine which die won.

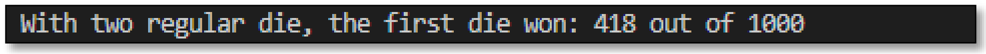
4. Comment out the previous display lines. They were only used for debugging.

5. Create a loop to roll the dice 1000 times.

    a. Accumulate the number of times the first **Die** object has a higher value than the second **Die object**.

    b. Display the results.

Example run:



```
With two regular die, the first die won: 418 out of 1000
```

## Part 3: Loaded Dice

Create a **LoadedDie** class that can give a player a slight advantage over the computer.

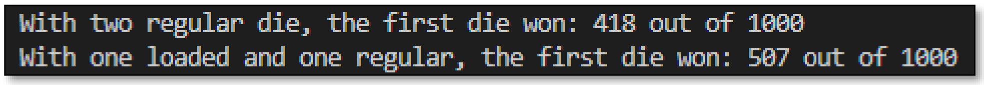
1. Copy the Die class header and implementation files. Rename them.

2. A **LoadedDie** never rolls a 1; it only rolls values 2 through 6.

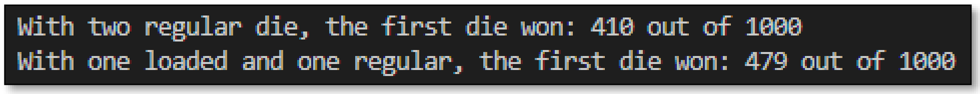
## Part 4: Main Program

1. Create an application that rolls two **Die** objects against each other 1,000 times.
    a. Accumulate the number of times the first **Die** object has a higher value than the second **Die object**.
    b. Roll a **Die** object against a **LoadedDie** object 1,000 times and count the number of times the first **Die** wins.
    c. Display the results of each as shown.

Example application run:



```
With two regular die, the first die won: 418 out of 1000
With one loaded and one regular, the first die won: 507 out of 1000
```



```
With two regular die, the first die won: 410 out of 1000
With one loaded and one regular, the first die won: 479 out of 1000
```

Each run will be different. They will be relatively close to the example.

## Assignment Submission

1. Attach the pseudocode.
2. Attach the program files.
3. Attach screenshots showing the successful operation of the program.
4. Submit in Blackboard.