

# Program #7: Let's Play Craps!

**Due Date: July 18, 2000**

## 1 The Problem

Craps is a game played with a pair of dice. In the game of craps, the *shooter* (the player with the dice) rolls a pair of dice and the number of spots showing on the two upward faces are added up. If the opening roll (called the 'coming out roll') is a 7 or 11, the shooter wins the game. If the opening roll results in a 2 (snake eyes), 3 or 12 (box cars), the shooter loses, otherwise known as 'crapping out'. If the shooter rolls a 4, 5, 6, 8, 9 or 10 on the opening roll, then he or she must roll the same number before rolling a 7 to win the game. For example, if the shooter rolls a 6 on the come out roll, a 10 on the second roll and a 7 on the third roll, the shooter loses since he rolled a 7 before rolling another 6. If, however, he rolled a 6 on the third roll, he wins the game.

## 2 The Program

In this assignment, you will write a program to play a certain number of craps games and then print out statistics on the games played. You will use *three* classes to implement the craps games, class `Dice`, `CrapsGame` and `PlayCraps`. The main method will be contained in class `PlayCraps`.

### 2.1 The class `Dice`

The purpose of this class is to establish the structure (instance variables) of a pair of dice and specify the operations (methods) that can be performed

on a pair of dice. A *partial* declaration of the class `Dice` is given below.

```
class Dice {  
  
    private int  
        face1,                //number on face 1  
        face2;                //number on face 2  
  
    private static int getRandInt1To6() { //private method to  
        return (int) (6*Math.random() + 1); //return a random  
    } //integer in the  
                                           //range from 1 to 6  
  
    //more methods here  
  
}
```

The declaration has a private, static method called `getRandInt1To6` (short for ‘get random integer from 1 to 6’) which is coded for you. This method returns a ‘random’ integer in the range from 1 to 6. That is, it returns a number in the specified range with *equal probability*, namely, each integer between 1 and 6 has probability  $1/6$  of being selected. (Random numbers are needed in game playing programs. Without randomness built into these programs, the computer would play the same game over and over again.) Note that the structure of a pair of dice can be specified by a pair of integers, `face1` and `face2`.

To implement the remainder of the class declaration, complete the coding of the following constructors and methods.

- `public Dice();` //a constructor that creates a pair of dice and sets both `face1` and `face2` to `-1`
- `public int getSumOfFaces();` //a method that returns the sum of the two faces
- `public void roll();` //a method that performs one roll of the dice

The method `roll` should call the private method `getRandInt1To6`.

## 2.2 The class CrapsGame

This class uses the class `Dice` to describe the structure of a game of craps as well as the operations that can be performed on a game of craps. A *partial* declaration appears below. Notice that a game of craps consists of a pair of dice (`pairOfDice`) along with two variables `win` and `numRolls`. The variable `win` is of type `boolean` and `numRolls` is of type `int`. The variable `pairOfDice` is of type `Dice`.

```
class CrapsGame {  
  
    private Dice  
        pairOfDice;        //a pair of dice  
  
    private boolean  
        win;                //equals true if game  
                            //a win, false otherwise  
  
    private int  
        numRolls;          //total number of rolls  
  
    //more methods here  
  
}
```

To implement the remainder of the class declaration, complete the coding of the following constructors and methods.

- `public CrapsGame();` //a constructor that creates a craps game and sets `win` to `false`, `numRolls` to 0 and creates a `pairOfDice`
- `public void reset();` //sets `win` back to `false` and `numRolls` back to 0, that is, resets a game of craps
- `public void play();` //plays a game of craps
- `public boolean getWin();` //returns the value of `win`
- `public int getNumRolls();` //returns the value of `numRolls`

Method `play` is the method requiring the most thought. The others are rather trivial to code. To get started on coding method `play`, consider the following pseudocode.

```
public void play() {
    roll the dice;
    increment number of rolls;
    firstRoll = pairOfDice.getSumOfFaces();
    if (firstRoll was a 7 or 11)
        game is won;
    else if (firstRoll is not snakeeyes, not a three and
            not boxcars) {
        do {
            roll the dice;
            increment number of rolls;
            value = pairOfDice.getSumOfFaces();
        }
        while (value is not firstRoll or 7);
        if (value is firstRoll)
            the game is won;
    }
}
```

Note that no where in the pseudocode is the game ever declared lost. The reason is that when a `CrapsGame` object is created or reset, the instance variable `win` is set to `false`. Therefore, a game is assumed lost unless the player wins it!

### 2.3 The class `PlayCraps`

Write a class called `PlayCraps` (the demo class) that prompts the user to input a positive integer representing the number of craps games to be played. The program then plays the required number of games and prints out the following statistics.

- number of games played
- number of wins
- length of the longest game played

- estimated probability of winning at craps expressed as a decimal between 0 and 1
- total number of rolls made
- average number of rolls per game expressed as a decimal
- number of wins that occurred on the coming out roll
- estimated probability of winning a game on the coming out roll expressed as a decimal between 0 and 1
- number of losses that occurred on the coming out roll
- estimated probability of losing a game on the coming out roll expressed as a decimal between 0 and 1

The statistics should be printed out in a *readable and attractive manner!* To get fairly good estimates of the probabilities of winning, etc., you should run a *large number* of games, say 10,000 or so. (You should start at 10,000 and then increase the number of games until your computer takes too much time to run them all.) Use the following formula to compute the estimated probability of winning at craps.

$$\text{probability of winning at craps} = \frac{\text{number of games won}}{\text{total number of games played}}$$

The other formulas you need are similarly obtained.

Your class `PlayCraps` might begin as follows.

```
class PlayCraps {

    public static void main (String[] args) {

        CrapsGame game = new CrapsGame(); //create a game of craps
        int numGames; //number of games to be played

        //other variable declarations

        System.out.println("Welcome to Craps!");
        System.out.println("\nEnter number of games: ");
        numGames = SavitchIn.readLineInt();

        //etc.

    }
}
```

Your classes may include other local variable declarations as needed. In addition, you may decide to implement other methods in your classes. For example, you might choose to invoke a static method to print out all the statistics.

### 3 What To Turn In

Please submit your `.java` source files in `e:\submit\DelGreco\Comp170\Prog7\<yourFolder>`. There will be three files in all: `Dice.java`, `CrapsGame.java` and `PlayCraps.java` (contains the main method).

### 4 Late Program Policy

A program will lose 10% of its value each day it is late (excluding weekends). *Starting early on your programs will maximize your chances of earning full credit on your work!*