**CIS 162 Project 3**

**A Game of Craps**

**(ALL games/projects assigned in CIS162 are designed to help students understand basic Java programming and have some fun learning some of these techniques. This project is for educational purposes only and can only be used in CIS162.**

**Due Date**

* at the start of lab on Thursday

**Before Starting the Project**

* Read chapter 5
* Read this entire project description before starting

**Learning Objectives**

After completing this project you should be able to:

* *create* a Graphical User Interface (GUI)
* write methods to meet specific requirements
* *write* conditional statements with Boolean expressions

**Game Rules**

A game starts with 10 credits. A player's turn includes one or more rolls of both dice until she either wins or loses. The player rolls two dice to start a turn. This is called the "come out" roll. The player wins 1 credit if the come out is 7 or 11. The player loses 1 credit if the come out is 2, 3 or 12. Otherwise, the "point" is set at the value of the come out. The player continues to roll until the "point" is rolled again (the player wins 1 credit) or a 7 is rolled (the player loses 1 credit). A new turn begins until the player runs out of credits or chooses to stop.

**Step 1: Create a New BlueJ Project**

**Step 2: Class Definition: GVdie**

Rather than writing your own Die class, we are providing a completed class for you. Create a new class in BlueJ called GVdie and delete all of the provided code. Cut and paste the provided code from (GVdie.java) into the newly created class. It should compile with no errors.

**Step 3: Create a class called Craps (70 pts)**

* Provide appropriate names and data types for each of the instance variables. Maintain two GVdie objects, current point, credit balance, current message and a boolean value that is set to 'true' when it is legal to come out and set to 'false' when it is not.
* public Craps ( ) - a constructor that initializes all of the instance variables to appropriate values.
* public void comeOut ( ) - If it is time to come out, roll both dice. Otherwise, remind the player by setting the message instance variable that it is not time to “come out” but it is time to roll (do not print a message). If the total is 7 or 11, the player wins and an appropriate message is set. If the total is 2, 3 or 12, the player loses and an appropriate message is set. Otherwise, the dice total is set to the "point". The player must have at least one credit to play. Update instance variables as needed.
* public void roll ( ) - If a point has been set, roll both dice. Otherwise, remind the player to come out first by setting the message. If the total is 7, the player loses and an appropriate message is set (not printed). If the value of the rolled dice equals the "point", the player wins and an appropriate message is set (not printed). Update instance variables as needed.
* public int getCredits ( ) - return the current credit balance.
* public boolean okToRoll ( ) – return true if it is time to roll, return false if it is time to come out.
* public void setCredits (int) - set the credit balance to the provided value. The value must be positive or else the credit remains unchanged.
* public GVdie getDie (int num) - return the requested die. Legal values for "num" are 1 or 2. Note, this method is only used for the GUI.
* public String getMessage ( ) - return the internal message. The message takes on different meanings depending on what is happening in the game.

**Step 4: Software Testing**

Software developers must plan from the beginning that their solution is correct. BlueJ allows you to instantiate objects and invoke individual methods. You can carefully check each method and compare actual results with expected results. However, this gets tedious. Another approach is to write a main method that calls all the other methods. See Listings 4.1 and 4.3.

**Main Method (5 pts)**

Write a main method in the Craps class to automatically play one game. You can then invoke the method repeatedly to test all conditions. This will require a simple while loop (Section 5.4)

1. Instantiate a Craps object
2. Print the initial message
3. Come out
4. Print the current message
5. Continue as long as it is valid to Roll
   1. Roll
   2. Print the current message
6. Print the number of credits after the game is over

**Sample Results**  
The following sample shows the results of three games. Your messages can be more creative as long as they convey the necessary information.

Welcome to Scott's Game of Craps!

Come out.

You rolled 4 and your point is 8. Roll again.

You rolled 9 and your point is 8. Roll again.

You rolled 6 and your point is 8. Roll again.

You rolled 8 and win!

Come out.

You rolled 7 and win!

Come out.

You rolled 12 and crapped out.

**Test Plan (5 pts)**

Games and other applications that generate random results are difficult to test. One useful strategy is to generate a checklist for you, or someone else, to confirm each possible result is correct. Create a test plan with a dozen or more test cases. Think of this test plan as a checklist of game possibilities. You should continue to play the game until all test cases have occurred and you have confirmed accurate results. Provide your plan along with the source code for this project. Here are TWO sample test cases to get you started but your test plan will likely have at least a dozen or more. The test plan will be graded based on how thorough it is.

Test Cases

\_\_\_ If initial roll is 7, did player win? Did credits increase by one?

\_\_\_ If initial roll is 11, did player win? Did credits increase by one?

**Step 4: Create a GUI class (10 pts)**

Now that you have the basic game working within it’s own class it is time to create a more interesting graphical user interface for the player to use. Create a new class that extends JPanel (see Listing 5.13 for an example).

* Define instance variables to hold two GVdie, a Craps object, a JLabel to display the game message, a JLabel to display the current point total, a JButton to come out and a JButton to roll.
* public GUIpanel() – instantiate both JLabels and both JButtons. Instantiate the Craps object and display the initial message in the JLabel. Define but do not instantiate two GVdie. Instead, assign values to the two GVdie instance variables by invoking the Crap’s getDie( ) method twice. Your solution will look a bit different but here is an example:

guiDie1 = myGame.getDie(1);

guiDie2 = myGame.getDie(2);

**Inner Class for the Action Listener**

Create an inner class within the GUIpanel class that implements ActionListener (see the example in Listing 5.13). Depending on which of the two buttons are pressed, either invoke Crap object’s roll() or comeOut() method. In both cases, update the message.

**Main method**

Create a main method that instantiates a JFrame and displays a GUIpanel. See Listing 5.12 for an example. This method will only have five lines of code. Invoke this method to play the game.

**Challenge Requirements (10 pts)**

The following should only be attempted after all of the other requirements have been completed. You are encouraged to attempt both but you receive points for only one.

Use the setEnabled feature of the JButton class to gray out the button if it is not legal to either Come Out or Roll. Only one button at a time will be enabled. You are on your own to find out how to use this method.



or

Use BorderLayout described in Section 7.11 for a more attractive layout. Place the message in the North region, both dice in the Center region and the remaining components in the South region. Hint: you must create an additional JPanel to hold the buttons and credits before placing it in the South region. The same is true for the two dice in the center. Your solution does not have to look exactly like this example.

****

**Grading Criteria**

There is a 50% penalty on programming projects if your solution does not compile.

* Stapled cover page with your name and signed pledge. (-5 pts if missing)
* Project requirements as specified above. (85 pts)
* Elegant source code that follows the GVSU [Java Style Guide](http://www.cis.gvsu.edu/studentsupport/javaguide). (15 pts)

**Late Policy**

Projects are due at the START of the class period. However, you are encouraged to complete a project even if you must turn it in late.

* The first 24 hours (-20 pts)
* Each subsequent weekday is an additional -10 pts
* Weekends are free days and the maximum late penalty is 50 pts.

**Turn In**

A professional document **is stapled** with an attractive cover page. Do not expect the lab to have a working stapler!

* Cover page - Your project must have a cover page that includes your name, a title, an interesting graphic or photograph related to the project topic and the following signed pledge: "I pledge that this work is entirely mine, and mine alone (except for any code provided by my instructor). " You are responsible for understanding and adhering to the [School of CIS Guidelines for Academic Honesty](http://www.cis.gvsu.edu/Academics/Honesty/).
* Test Plan – a dozen or more test cases used to confirm that all aspects of the game work properly.
* Source code - a printout of your elegant source code for the Craps and GUI classes. You do not need to provide the GVdie class.