Final Project Evaluation Information for Intro to Java Programming

This document is intended for students enrolled in the full course for Intro to Java Programming. If you are using the course materials but are not enrolled, you are encouraged to complete the Final Project, but please understand that you will not submit your project for evaluation by the Udacity Coach team.

If you are enrolled in the full course, you can earn a verified certificate by successfully completing the Final Project according to the rubric included in this document and then passing an exit interview to authenticate your work on the project as your own.

In this document, you will find:

- **Rubric** - This is what your project evaluator will use to determine whether or not your Final Project meets our specifications.
- **Submission Instructions** - This will tell you how to submit your project to the Udacity Coach team.
- **Project Summary Questions** - You will include answers to these questions when you submit your project to the Udacity Coach team.

The Rubric

This rubric is here to help you understand the expectations for how your project will be evaluated. It is the same rubric that the person evaluating your project will use. You should look at the rubric **before you begin working** on this project **and before you submit it**.

How Evaluation Works

The purpose of the evaluation process is to give you feedback to help you to take your programming skills to the next level. Your project evaluator will read through and test your code for the project and compare it to the specifications listed in the rubric below. Your grade will simply be “meets specifications” or “does not meet specifications.”

- If you receive a “meets specifications” for **every** criterion listed in the rubric, then your project as a whole will receive a “meets specifications”.
- If you receive a “does not meet specifications” in **any** of the criteria in the rubric, then your project as a whole will receive a “does not meet specifications”. You will, however, be able to make changes and re-submit. Please note that it is perfectly normal to have to submit the project more than once.

How to Use the Rubric

- Once you think you have completed the project, go through each criteria item in the rubric below and do your best to honestly evaluate where you think your project falls.
- If you think your project “does not meet specifications” for **any** criteria item, then you should make changes accordingly.
- Once you’re confident that your project “meets specifications,” go ahead and follow the project submission instructions to submit!

Criteria	Does Not Meet Specifications	Meets Specifications	Exceeds Specifications (Completely Udacious)
Code Functionality			
The code compiles.	The code compiles but throws Exceptions or Errors. These include but are not limited to NullPointerExceptions, Compile errors, Runtime errors, etc.	The code compiles and runs without throwing any Exceptions or Errors. These include but are not limited to NullPointerExceptions, Compile errors, Runtime errors, etc.	*Not Applicable*
The agent has a functional move method.	The agent does not have a move method. The agent's move method does not help the agent decide on the move to be made based on the current game board. The agent's move method does not execute the moveOnColumn method. The agent's move method sometimes causes the agent to make illegal moves (e.g. skip its turn, add multiple pieces, move an earlier piece, or remove an earlier piece).	The agent has a move method. The agent's move method helps it decide on the move to be made based on the current game board. The agent's move method executes the moveOnColumn method. The agent's move method never allows the agent to make illegal moves.	*Not Applicable*
The agent recognizes and responds appropriately to available moves that would cause either player to win the game.	The agent does not correctly recognize when winning moves are available for itself and does not act to take advantage of those moves. Agent does not correctly recognize when winning moves are available for the opponent and does not block them accordingly.	The agent implements basic expected logic, such as the iCanWin() and theyCanWin() methods. Agent correctly takes winning moves when they're available and blocks opponents' winning moves when they would otherwise be available.	Agent's logic goes beyond the expected logic of playing winning moves and blocking opponents' winning moves. iCanWin() and theyCanWin() may be used, but other methods, such as evaluating hypothetical next moves or assessing long-term strategies, are used.
The agent can reliably win the game.	The agent does not beat the Random Agent at least 18 times out of 20 or the Beginner Agent at least 12 times out of 20.	The agent beats the Random Agent at least 18 times out of 20 and the Beginner Agent at least 12 times out of 20.	The agent beats one of the more difficult opponent agents (e.g. Brilliant Agent) at least 12 times out of 20.
Use of Control Flow Statements			

Appropriate control statements (for, while, if, else) are used in each relevant situation.	Selection of control statements is often inappropriate.	Selection of control statements is rarely inappropriate.	Selection of control statements is never inappropriate.
Iterative control statements (for, while) are used effectively to avoid repetitive code.	Iterative control statements are not always used where appropriate, causing unnecessary repetition in the code.	Iterative control statements are appropriately used to avoid repetition in the code.	*Not Applicable*
Conditional control statements (if, else) effectively used to guide code flow.	Conditional control statements are used incorrectly or in a way that is unnecessarily complicated.	Conditional control statements are used correctly and with clear logic.	Cleverly implements nested loops or nested conditions that make the code more efficient.
Definition and Use of Methods			
Repeated blocks of code are encapsulated in methods.	Blocks of code are used over and over again and not placed in their own methods instead.	Methods are defined to prevent repetitive blocks of code.	Repetitive blocks of code are appropriately contained within their own methods. Helper methods are defined effectively and contribute to code functionality and clarity.
Methods defined in the program are called where appropriate.	New code is sometimes written when a method already defined in the program could have been called instead.	Methods defined in the program are called where appropriate.	*Not Applicable*
Parameters and return values are appropriate for each method's purpose.	Choice of parameters and/or return values is not aligned with each method's purpose.	Choice of parameters and/or return values is aligned with each method's purpose.	*Not Applicable*
Use static and instance members of a class properly.	Not all usage of static or instance variables is correct.	All usage of static or instance variables is correct.	*Not Applicable*
Object Oriented Programming			
Accessors	Student code does not make use of any accessors.	Student code correctly uses accessor methods to get game data out of the board.	Student demonstrates proper conventions when using accessor methods, such as creating local references to content.
Mutators	Student code does not use mutators correctly.	Student code correctly uses mutators to modify the game board.	*Not Applicable*
Encapsulation	Student code ineffectively or inefficiently	Student code efficiently and effectively	*Not Applicable*

	draws information out of the object oriented class structure representing the game	leverages the object-oriented class structure of the game.	
Code Readability			
Naming conventions from the Java Language Coding Guidelines are used: • Variable and method names are lowercase, with occasional upperCase characters in the middle. • Class names start with an Uppercase letter. • Constant names are UPPERCASE, with an occasional UNDER_SCORE, and are declared as final.	One or more of the naming conventions outlined here from the Java Language Coding Guidelines are not followed consistently.	All of the naming conventions outlined here from the Java Language Coding Guidelines are followed consistently.	*Not Applicable*
Method documentation is clear and concise and is included for each method (see below for an example).	Javadoc comments are not included for each method or do not follow the conventions outlined in Lesson 3 of the course.	Javadoc comments are included for each method and follow the conventions outlined Lesson 3 of the course.	*Not Applicable*
Names of methods and parameters are descriptive and meaningful.	Names of some methods or parameters are not immediately understandable or do not reflect their purpose.	Names of all methods and parameters are immediately understandable and reflect their purpose.	*Not Applicable*
Magic numbers are avoided. `final` **variables are defined for numeric constants used in the program.**	Numeric constants (other than -1, 0, 1, and 2) are embedded in the code without a constant definition.	Numeric constants (other than -1, 0, 1, and 2) used in the code are defined as `final` variables.	*Not Applicable*

Below is an example of how to write javadoc for a method.

```
/**
   Convert calendar date into Julian day.
   Note: This algorithm is from Press et al., Numerical Recipes
   in C, 2nd ed., Cambridge University Press, 1992
   @param day day of the date to be converted
```

```
    @param month month of the date to be converted
    @param year year of the date to be converted
    @return the Julian day number that begins at noon of the
    given calendar date.
 */
public static int dat2jul(int day, int month, int year)

{
   . . .
}
```

Project Submission and Re-submission Instructions

Review the rubric above for the project. Iterate on your project until you feel that it meets our specifications.

When you feel confident that you have successfully completed the project, send an email to introjava-project@udacity.com containing your entire BlueJ project as a zip file and your answers to the Project Summary Questions below. You will receive a confirmation email from support@udacity.com that your message was received.

Within 7 business days, you will receive an email from your project evaluator (who will not be your Udacity Coach) with a completed evaluation and instructions for next steps.

If your project is evaluated as meeting our specifications, you will schedule an exit interview. The purpose of this interview is to talk to you about the project and to verify that you are the person who created the project you submitted. In preparation for this interview, please look back through your code for the project and make sure that you can explain the logic behind your work and your thought process in building it.

If your project is evaluated as not meeting our specifications, first of all, don't worry! The Final Project is itself a learning process, and This is an opportunity to further deepen your understanding of the course material and refine your work. Please reach out to your Coach through email or contact us through the blue chat box in the Intro to Java Programming classroom page if you have questions as you're making modifications to your project.

If you have any questions, please feel free to email your Coach!

Project Summary Questions

Please include answers to the following questions when you submit your project to the Udacity Coach team:

What is your name?

Joe Schmo

What E-mail address do you use to sign in to Udacity?

joeisthebest@example.com

You must include a list of Web sites, books, forums, blog posts, github repositories, etc. that you referred to or used in creating your submission (add N/A if you did not use any such resources).

List your citations here.

Please carefully read the following statement and include it in your email:

*"I hereby confirm that this submission is my work. I have cited above the origins of **any** parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc. By including this in my email, I understand that I will be expected to explain my work in a video call with a Udacity coach before I can receive my verified certificate."*

Is there any other important information that you would want your project evaluator to know?

Use this space to communicate with your project evaluator. Is there anything you would like to share? Feedback or suggestions?