Informatics 1: Object Oriented Programming

Assignment - Part II

The University of Edinburgh
Volker Seeker (volker.seeker@ed.ac.uk)
Samuel Macleod (s1712153@sms.ed.ac.uk)

1 Introduction

This is the second part of the assignment (PART II). Its aim is to help you understand that writing code is not just about functional correctness but also code quality. Well written code is easier to maintain, share and modify and writing it is an essential skill every developer should have. A common technique to improve code quality amongst professionals is a *code review*. The idea of a code review is for developers to go over code from other developers to spot errors, propose suggestions for improvement and to share knowledge within the development team.

In this part of the assignment, you will be doing a code review for submissions of the first part of the assignment from your fellow students. To reduce the bias towards your own solution, you will be given random pairs of submissions which you have to compare and provide feedback by considering three specific metrics: *Code Documentation*, *Code Structure* and *Usage of the Java Language*.

Not only will this give you an opportunity to see other students' approaches for PART I, you can also demonstrate your understanding of what makes good quality code. Furthermore, going through this process yourself, you will get a better understanding of how your own code will be evaluated at the end of the course to determine a final mark.

1.1 Administrative Information

Deadline

The assignment is due at **16:00 on 6 March 2020**.

Assessment and Feedback

This second part of the assignment is assessed for credit and makes up part of the final grade you will get for the INF1B course. To receive high marks for this part of the assessment, make sure your feedback is:

Constructive Is the review helpful and does it propose actionable steps to improve the corresponding code?

Specific Does the review point to specific things in the corresponding code?

Justified Does the review provide explanations and give reasons and arguments?

Kind Is the review kind and uses friendly language?

Those categories will be considered when your comments are assessed at the end of the year. All feedback you give is shared anonymously with the corresponding authors of the code after the submission deadline. Note that you are marked on the **feedback you give, not the feedback you receive** for this part of the assignment (see Appendix A for more details regarding the marking criteria for INF1B).

Submission

For PART II of the assignment, you will be using a specifically designed **Peer Feedback Framework**. It shows pairs of individual submissions, allows you to provide and submit feedback and view the feedback you received from others after the submission deadline. See Section 3 for more details on how you can use it.

☞ Please be aware that there is **no deadline extension** possible for this part of the assignment to allow a smooth progression of the course. If you miss the deadline, it will affect your final mark.

Support

Since PART II is marked for credit, you should **strictly** adhere to the *Good Scholarly Practice* (see below) when composing your feedback. Do not share your comments or which code you review with other students and compose all feedback on your own.

Nevertheless, you will have the possibility to get support should you need it. At any time, you are welcome to discuss the assignment with any course instructor. You can do that during teaching sessions or via the course's Piazza forum[1].

When posting on Piazza, please put all questions into the corresponding assignment folder. Feel free to discuss general techniques amongst each other unless you would reveal an answer. If in doubt about revealing an answer, please ask the instructors privately.

Lastly, as before please make good use of any internet or literature resource you consider helpful for solving the tasks (that, of course, includes all the course material released so far).

Good Scholarly Practice

Please remember the University requirement as regards all assessed work for credit. Details about this can be found at:

http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct

[1]https://piazza.com/ed.ac.uk/spring2020/inf1b

Furthermore, you are required to take reasonable measures to protect your assessed work from unauthorised access. For example, if you put any such work on a public repository then you must set access permissions appropriately (generally permitting access only to yourself, or your group in the case of group practicals).

2 Tasks

For your code review, you should always compare two submissions with each other and comment on one or more of the three metrics listed below. Try to find code examples for which one submission shows better code quality compared to the other, describe them and give justifications as to why and how the inferior version could be improved[2].

You do not need to comment on all three metrics for one submission pair. It might be difficult to find a superior version regarding a specific metric for a given pair. For example, both version might not comment their code at all or both version might present extensive and helpful comments for all their methods and classes. If you find it difficult to give sound advice, you can generate a new pair and give feedback on comments for this one.

Please use your feedback to not only give helpful and actionable comments but also demonstrate your understanding of the concepts involved. We do not expect you to write much more than 300 to 400 words for each task.

✎ The focus of assignment PART II is on **Code Quality**, not on *Correctness and Robustness* of the submitted code. You can review auto marker feedback for each submission but you should not necessarily have to consider it for your comments.

Task 1 - Code Documentation

◁ **Task**

For this task you should find an example pair where one solution is better documented than the other and explain why. To do this, you can consider the documentation of classes and methods as well as in-line comments supporting the implementation. As a guideline, consider previously released *Inf1B Coding Conventions* and ask yourself the following questions:

- Do more comments generally improve code readability or obscure it?
- Are the comments useful to improve readability?
- Which method and class documentation comments help you understand what the corresponding components do and how you should use them without actually reading the code?
- Are all aspects of a method documented using the correct Javadoc tags, e.g. @param, @return, @throws, etc?
- Can you get an idea of what variables and methods should be used for from their name alone, i.e. are identifiers written in a self-documenting style?
- Is the code formatting clear and consistent?

[2] When you later review the feedback others gave you, you can always see which submission you were compared against.

Task 2 - Code Structure ◃ Task

For this task, you should find an example pair where one demonstrates a higher quality in code structure in their submitted solution compared to the other and explain why. To do this, you can consider the abstraction of various sub tasks using for example helper methods, length of methods, code duplication, consistency in ordering individual elements of a class, etc. As a guideline, ask yourself the following questions:

- How easily can you understand what the code is doing when reading it?
- If you wanted to modify it, how quickly could you find out where to do that?
- Can specific functionality be swapped out or updated easily without affecting calling code?
- Can specific functionality easily be used in other modules or programs?
- Is code duplication avoided where possible?
- Are constant values used instead of "magic numbers"?

Task 3 - Usage of the Java Language ◃ Task

For this task, you should find an example pair where one demonstrates a better use of Java libraries and language constructs in their solution compared to the other and explain why. To do this, you can consider the following questions as a guideline:

- Can parts of the code be swapped out for specific library functions or classes which already do the same job?
- Are library functions and classes used in the way they are intended?
- Are control statements such as loops and conditionals used in the way they should be for a Java program?
- Are exceptions and exception handlers used in the way they are intended for the Java language?

☞ Keep in mind, that the first part of the assignment was written under specific restrictions which will not be the case for part three. You can certainly use collection classes or a more intelligent object oriented structure to solve PART I in a better way. We will cover that in a later tutorial. For your feedback here, please focus on the use of available and allowed Java libraries.

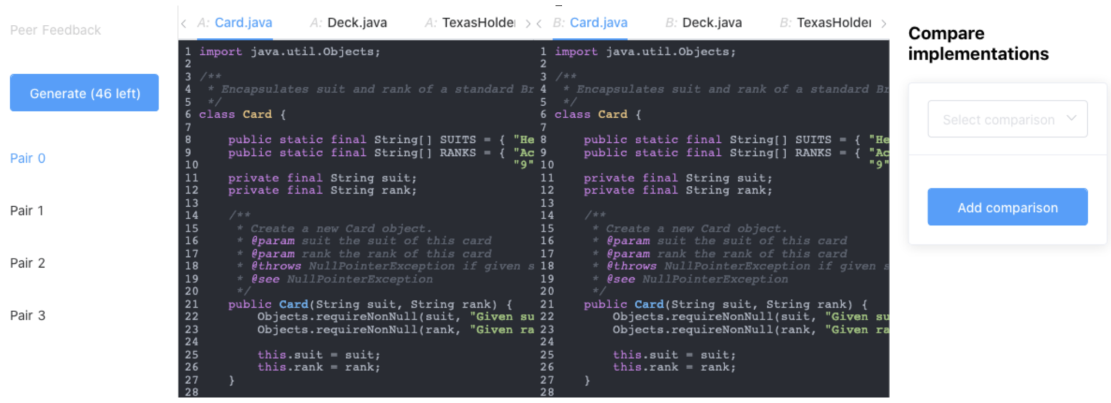
3 Peer Feedback Framework

The following section contains a description of the Peer Feedback Framework[3] we use for the INF1B course and how you can work with it. You will only be able to access this website if you have an Informatics DICE account.

Generating Review Pairs

You will be reviewing your colleagues' work in pairs, and comparing the Readability, Code Structure, and Java Usage between two student code submissions. To generate a new pair of code submissions for comparison, click the blue Generate button. There is a limit of pairs you can generate, which is shown on the button.

✎ Each displayed pair is anonymous; while you will be reviewing real student code, there is no way of identifying which student wrote it unless the student added author information to their source code.



```
import java.util.Objects;

/**
 * Encapsulates suit and rank of a standard Br
 */
class Card {

    public static final String[] SUITS = { "He
    public static final String[] RANKS = { "Ac
                                           "9"
    private final String suit;
    private final String rank;

    /**
     * Create a new Card object.
     * @param suit the suit of this card
     * @param rank the rank of this card
     * @throws NullPointerException if given s
     * @see NullPointerException
     */
    public Card(String suit, String rank) {
        Objects.requireNonNull(suit, "Given su
        Objects.requireNonNull(rank, "Given ra

        this.suit = suit;
        this.rank = rank;
    }
```

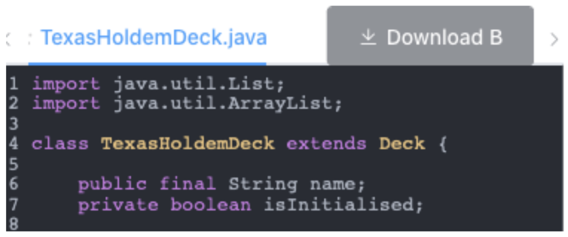
```
import java.util.Objects;

/**
 * Encapsulates suit and rank of a standard Br
 */
class Card {

    public static final String[] SUITS = { "He
    public static final String[] RANKS = { "Ac
                                           "9"
    private final String suit;
    private final String rank;

    /**
     * Create a new Card object.
     * @param suit the suit of this card
     * @param rank the rank of this card
     * @throws NullPointerException if given s
     * @see NullPointerException
     */
    public Card(String suit, String rank) {
        Objects.requireNonNull(suit, "Given su
        Objects.requireNonNull(rank, "Given ra

        this.suit = suit;
        this.rank = rank;
    }
```

Figure 1: Main view of the Peer Feedback Interface

Comparing Code

Click on the pair of code submissions you would like to review from the sidebar (if you have already compared this pair on some metric before, it will appear in bold). You will see the screen displayed in Figure 1, which has one code submission on the left (**Student A**), and the other on the right (**Student B**), each from different students. As you would expect, you can look through the different files of each submission using the two tab bars.

✎ Among other things, your comments need to be specific enough to be acted upon. In that sense you should ensure that you specifically refer to **Student A** and **Student B** when discussing two submissions in your code review.



```
import java.util.List;
import java.util.ArrayList;

class TexasHoldemDeck extends Deck {

    public final String name;
    private boolean isInitialised;
```

Figure 2: Downloading code samples

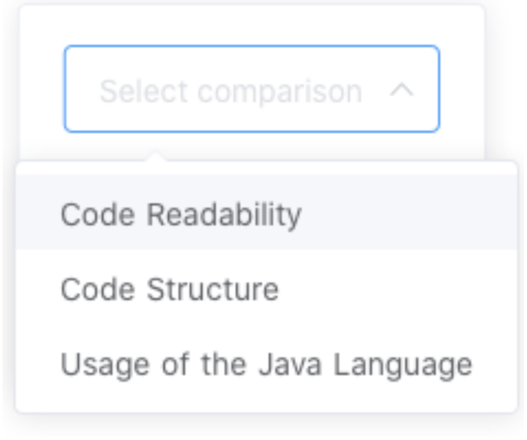


Figure 3: Metric Drop-Down

Downloading Code

If you would like to download one of the submissions (to run it on your machine, for instance), the last tab in each tab bar is a download button, which when clicked will download a zip file of that students submission onto your computer (see Figure 2). Each submission also contains a file *autograded.md* with corresponding CODEGRADE auto grader results.

Writing Feedback

Once you found a pair of code samples you would like to comment on for one of the three metrics, you can select the pair from the left sidebar and pick the corresponding metric from the drop-down on the right (see Figure 3). Press the Add comparison button to start writing your feedback. A rich text box will appear where you can write the substance of your comparison (see Figure 4). You can use normal formatting like bold and italics, as well as insert images or code snippets. Everything you write is auto-saved, so dont worry about losing your work.

✎ You can only submit one comparison per metric, and so if you have already commented on another pair for a specific metric, that metric will appear grayed out in the list.

☞ At the deadline for this part of the assignment, the comments you have written will be submitted automatically. So please make sure, you have everything in place like you want it.

Focus Mode

If you want more space to write, click Expand, which will expand the text box to fill your screen.

[3] https://groups.inf.ed.ac.uk/inf1b-op/

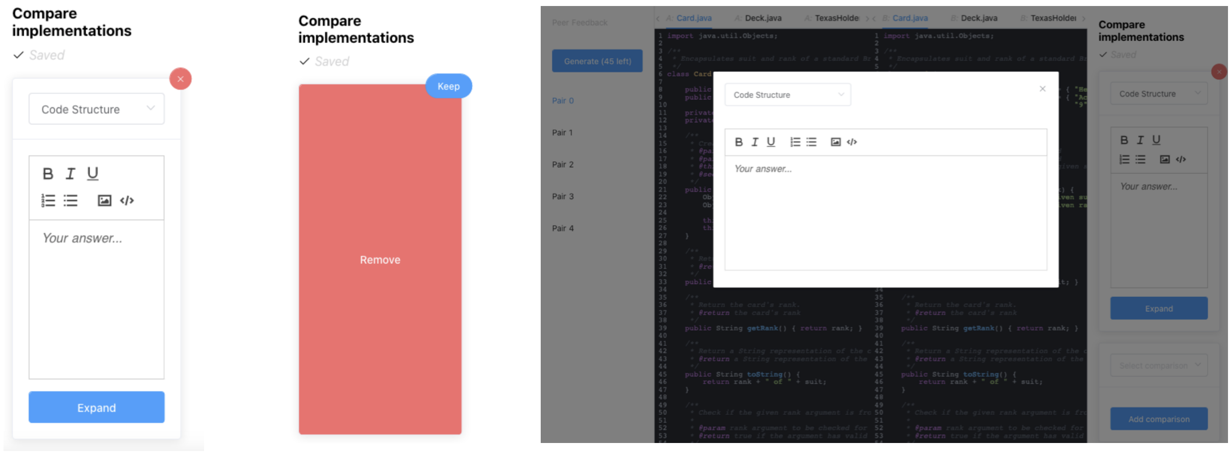


Figure 4: Regular and expanded input view and removing input.

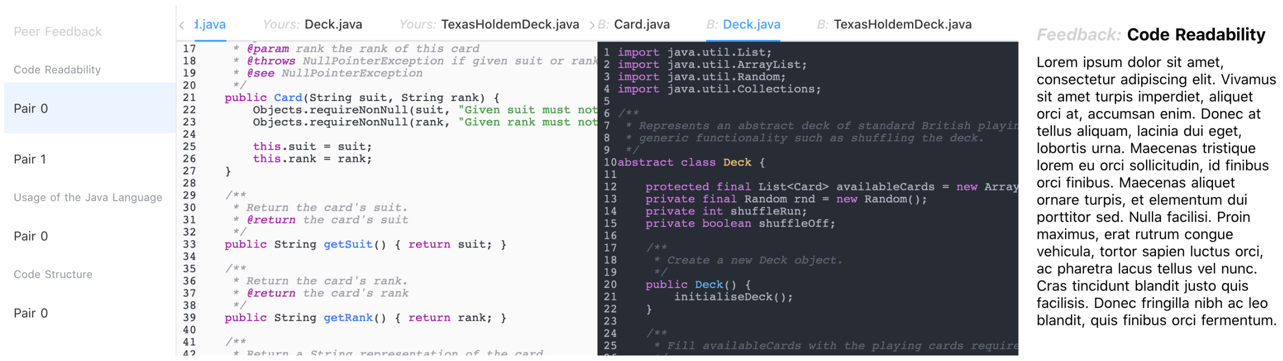
Removing Feedback

If you decide to remove a feedback entry - for instance if you find a pair that is more suitable to compare on that specific metric - just click the red $\boxed{\text{X}}$ button on the comment you would like to delete. This will give you a confirmation to make sure you want to remove it, click anywhere in the red area to proceed (see Figure 4).

☞ This process is irreversible and deleted comments are lost. If you want to be on the safe side, we suggest you copy and paste them into a document on your own machine first.

Reviewing Feedback from Others

After the submission deadline for this part of the assignment has passed, comments provided by other students for your submission of PART I will be made available to you via the same web interface. You will be able to select feedback from other students (which will also be anonymous) on the left and see which submission your own was compared against (see Figure 5). The tutorial sessions in the week following the deadline are intended for you to discuss comments with the tutors should you have any questions.



```
     * @param rank the rank of this card
     * @throws NullPointerException if given suit or rank
     * @see NullPointerException
     */
    public Card(String suit, String rank) {
        Objects.requireNonNull(suit, "Given suit must not
        Objects.requireNonNull(rank, "Given rank must not

        this.suit = suit;
        this.rank = rank;
    }

    /**
     * Return the card's suit.
     * @return the card's suit
     */
    public String getSuit() { return suit; }

    /**
     * Return the card's rank.
     * @return the card's rank
     */
    public String getRank() { return rank; }

    /**
     * Return a String representation of the card
```

```
import java.util.List;
import java.util.ArrayList;
import java.util.Random;
import java.util.Collections;

/**
 * Represents an abstract deck of standard British playin
 * generic functionality such as shuffling the deck.
 */
abstract class Deck {

    protected final List<Card> availableCards = new Array
    private final Random rnd = new Random();
    private int shuffleRun;
    private boolean shuffleOff;

    /**
     * Create a new Deck object.
     */
    public Deck() {
        initialiseDeck();
    }

    /**
     * Fill availableCards with the playing cards require
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus sit amet turpis imperdiet, aliquet orci at, accumsan enim. Donec at tellus aliquam, lacinia dui eget, lobortis urna. Maecenas tristique lorem eu orci sollicitudin, id finibus orci finibus. Maecenas aliquet ornare turpis, et elementum dui porttitor sed. Nulla facilisi. Proin maximus, erat rutrum congue vehicula, tortor sapien luctus orci, ac pharetra lacus tellus vel nunc. Cras tincidunt blandit justo quis facilisis. Donec fringilla nibh ac leo blandit, quis finibus orci fermentum.

Figure 5: Feedback view available after submission deadline

Appendix A Inf1B Marking Criteria

This section contains a description of the assessment scheme used to determine the overall course mark for each student at the end of the semester. This happens after you have submitted the third part of your assignment at the end of week 11. For this process we will consider the feedback you generated for PART II and your code submission for PART III.

We will consider the following criteria:

Completion Those with less previous experience may have difficulty completing all of the assignment tasks. It is possible to pass without attempting the more advanced tasks - and a good solution to some of the tasks is much better than a poor solution to all of them.

Readability & Code Structure Code is a language for expressing your ideas - both to the computer, and to other humans. Code which is not clear to read is not useful, so this is an essential requirement.

Correctness & Robustness Good code will produce "correct" results, for all meaningful input. But, it will also behave reasonably when it receives unexpected input.

Use of the Java Language Appropriate use of specific features of the Java language will make the code more readable and robust. This includes, for example: iterators, container classes, enum types, etc. But the structure of the code, including the control flow, and the choice of methods, is equally important.

Code Review Working with code provided by other developers is a major part of working with software. The ability to assess a piece of code written by someone else and to provide meaningful feedback on its quality is therefore an essential skill you should learn.

Marks will be assigned according to the University Common Marking Scheme[4].

The following table shows the requirements for each grade. All of the requirements specified for each grade must normally be satisfied in order to obtain that grade.

Pass: 40-49%

- Submit some plausible code for a significant part of assignment PART III, even if it does not work correctly.
- Demonstrate some understanding of the issues involved in your answers to at least one question in the Code Review from PART II.

Good: 50-59%

- Submit working code for most parts of assignment PART III, even if there are small bugs or omissions.

[4] https://web.inf.ed.ac.uk/infweb/student-services/ito/students/common-marking-scheme

- Submit code which is sufficiently well-structured and documented to be comprehensible. This includes an appropriate use of Java features and choice of methods, as well as layout, comments and naming.
- Demonstrate some understanding of the issues involved in your answer to all of the questions in the Code Review from PART II.

Very Good: 60-69%

- Submit working code for all parts of assignment PART III, even if this contains small bugs.
- Submit code which is well-structured and documented.
- Demonstrate good understanding of the issues involved in your answers to all of the questions in the Code Review from PART II and provide plausible and actionable solutions for some.

Excellent: 70-79%

- Submit and demonstrate working code for all parts of assignment PART III, with no significant bugs.
- Submit code which is well-structured and documented.
- Demonstrate good understanding of the issues involved in your answers to all of the questions in the Code Review from PART II and provide plausible and actionable solutions for all of them.

Excellent: 80-89%

- Marks in this range are uncommon. This requires faultless, professional-quality design and implementation, in addition to well-reasoned and presented answers to the Code Review questions.

Outstanding: 90-100%

- Marks in this range are exceptionally rare. This requires work "well beyond that expected".