Informatics 1: Object Oriented Programming
Tutorial 03 - Code Golf

Brian Mitchell
Vidminas Vizgirda
Adriana Sejfia

Week 7

1 Introduction

In this tutorial, we will practice skills such as understanding problem scenarios, designing programs, and understanding the relationships between code and design. This is partly why there is a mandatory reflective task at the end. This tutorial is also about seeing how programs are built from small pieces. It gives you practice for Assignment 2 and includes practical skills such as using markdown text formatting and using zip files for IntelliJ projects.

2 ASCII Dice

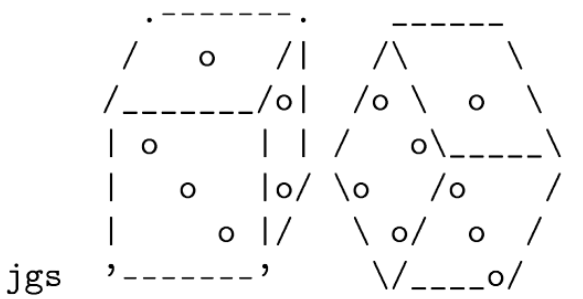

Figure 1: ASCII Art Dice by Joan G. Stark. Copyright © 1996-01 Joan G. Stark

Your task will be designing, coding, and analysing a program to "pretty print" the faces of a standard 6-sided die. You can imagine this as part of a larger project such as a digital board game. You will also look at how the design and code are explained to beginners—this aspect is directly relevant to the Inf1B assignments. "Pretty printing" means displaying text in a fancy or elaborate way. For this task, our goal will be to take a digit from 1 to 6 (the roll of a die) and print a picture using plain text, a technique called ASCII art. The output should resemble the dots on the appropriate face of the die. You will not be drawing

complex art like the above illustration. Instead, it will be enough to draw the top-down view of a rolled die face, using spaces, O characters, dashes, and pipe symbols, like below:

```
1:    2:     3:   4:    5:    6:
-----  -----  -----  -----  -----  -----
|   |  |O  |  |O  |  |O O|  |O O|  |O O|
| O |  |   |  | O |  |   |  | O |  |O O|
|   |  |  O|  | O |  |O O|  |O O|  |O O|
-----  -----  -----  -----  -----  -----
```

3 Tasks

3.1 Task 1 - Team up

Pair up with a partner to work with during this tutorial. If you have an odd number of people, you can also work in a group of 3. Like in previous tutorials, you should use pair programming, switching between driver and navigator roles. The tutors will no longer prompt you when to switch, so you should decide when you are comfortable to do this yourselves but make sure that everyone in your pair (or group of 3) gets a chance to spend sometime in both roles.

3.2 Task 2 - Download and open the project design and code template

Download the ASCIIDice.zip archive from the course materials site (where you found these tutorial instructions). Instructions for opening the project and navigating the menus:

1. in Explorer / Finder / Nautilus / other file manager: unzip the downloaded zip file (for Mac users, this might already be automatically done for you when you download the file)
2. optionally move the unzipped folder to somewhere more convenient for working
3. in IntelliJ: Open a new project, navigate to the directory containing the ASCIIDice.iml file BUT open the directory NOT the .iml file.
4. IntelliJ will now open the project but the IntelliJ window doesn't show you much initially, which can make you think nothing has happened, so:
5. open the Project view (Alt+1 or click on the "Project" tab in the top-left of the screen with sideways writing)
6. open the Structure view (Alt+7 or click on the "Structure" tab in the bottom-left of the screen with sideways writing)
7. in the Project view double-click on readme.md to open the project description file
8. in the Project view, expand the src folder to see the provided source code. You should see one file: PrintDice.java
9. at the bottom of the IntelliJ screen, click on the "TODO" tab. It will show you all comments in the project code starting with "TODO". You can click on items in the "TODO" tab to navigate to them in source code.

3.3 Task 3 - Read the introduction to Markdown

Markdown is a widely used standard for marking up text files in a simple way. Although markdown only uses basic text characters it allows font changes, colour, automatically formatted code listings, automatically formatted tables, footnotes, hyperlinks, and graphics (embedded via a hyperlink). You can use markdown on Piazza which will give you an opportunity to practice.

There are different styles of markdown but as long as it works in IntelliJ's viewer and is consistent, that is acceptable. You need to use markdown for Assignments 2 and this is just an introductory glance.

In the project, open the markdown.md file. Take about 5 minutes to read its contents. The file has instructions for how to view markdown in editor and preview modes – try out the available options.

Complete the IntelliJ configuration steps.

Skip the further reading links for now – you can come back to these after the tutorial, if you want.

3.4 Task 4 - Read the project README file

The readme.md file walks through the setup of this project, some hints, and steps to implement an ASCII Dice printing program.

Read these instructions and clarify any confusing parts. You might want to draw on paper or otherwise visualise how this program will work.

If you're unsure about something and none of the partners in your group know, try finding the answer on the internet, and if you still can't find it, then ask a tutor for help. You might need around 10-15 minutes for this.

3.5 Task 5 - Implement and test code

Now that you have the design in mind, write code to implement it. You should populate the provided PrintDice.java file and add additional files (if needed). If you feel confident in writing the code, you can make improvements the proposed design, but this is optional.

Take your time in writing code. If you get stuck, don't hesitate to ask for help. If you finish quickly, move on to the next tasks, but if you need more time – that's okay, just leave at least 20 minutes for the writing, reflection, and exporting tasks before the end of the tutorial, but you can skip the rest.

3.6 Task 6 - Write up the program design

Open the walkthrough.md file. It contains a design write-up template. Take at least 10 minutes to fill in this template, describing, in your own words, how your implementation works. Write for a beginner audience.

3.7 Task 7 - Write reflections

Reflecting on each tutorial (and lab sheet) is important to strengthen your experience and skills. It helps reinforce your learning and can reduce the amount of revision needed in future. Your notes can be about anything you feel is important but should probably include:

- any unexpected difficulties;
- any difficulties you expected that did not occur;
- any useful knowledge you gained;
- any existing knowledge you reused;

- any connections you have made.

Open the reflections.md file and add your reflections there.

Take at least 10 minutes to answer the questions and prompts in the provided template. Also try to think how each item might be useful to you in future, even if you do not yet understand it fully.

3.8 Task 8 - Export your work

For Assignment 2 you need to submit an IntelliJ project that has been properly exported as a zip file. You can also practice how to do that for this tutorial's project.

1. "Build > Rebuild Project" and fix any compilation errors — in assignments, code which does not compile (for any reason, no matter how trivial) scores 0.
2. Make no more changes — if you do return to step 1.
3. "File > Export > Project to Zip File ...". If you don't see this option, ensure you have the "Project to Zip" plugin installed and enabled in Intellij.
4. Ensure you do not include a previously exported zip file in the latest project export (a zip inside a zip) because this means if an autotester is used on your code it is not guaranteed to run the latest version of your code.
5. Copy your zip file somewhere else, unzip it, and check its contents are what you expected

That's it for the main part of the tutorial! All the steps you practiced here are very similar to what you will be required to do for Assignment 2. You can get a bit more practice with the optional tutorial tasks that follow. If you don't get to them during the tutorial, you can also complete them in your own free time. Feel free to ask for support on Piazza if you get stuck.

4 Optional: Code Golf and Microwave Timers

The first task was inspired by a "Code Golf" challenge: Convert numbers to dice patterns.

Code Golf is a website of coding challenges, where anyone can post challenges and solutions. Its name mirrors the game of golf where you win with the lowest number of strokes, except in code golf, you win by writing a solution with the lowest number of characters in your source code. Shortening of source code, often to extremes, is also sometimes called "minifying" or "uglifying". **However, we are not playing Code Golf with our work**. Though we do want you to eventually be able to write elegantly concise code, in Inf1B the most important things are that your code is readable and easy-to-understand – even if this means the code is not the shortest possible solution. Well-designed code is highly prized in both academic and corporate programming contexts

4.1 Task 1 - Another challenge

See this CodeGolf challenge page: Doing a little trolling with the microwave timer.

It will be the challenge of the next task. Read the challenge description and create a new IntelliJ project for this task.

4.2 Task 2 - Implement a solution

For fun, you could try re-implementing some of the proposed solutions from the challenge comments—some are possible to rewrite in Java. However, none of them are readable.

The real task is to implement a solution that works and is also easy to understand.

4.3 Task 3 - Describe the design

Copy the walkthrough.md file from the previous project and write up the design of your new implementation.

4.4 Task 4 - Reflect

Copy the reflections.md file from the previous project and add any new reflections you have after completing the above tasks.

4.5 Task 5 - Export your work

Follow the instructions from the previous task to export your work to a zip archive.