ECS414U/A - Object Oriented Programming
Miniproject instructions

Lecturers: Bruno Ordozgoiti & Matthew Huntbach

You must write a program from scratch. The purpose of this project is for you to demonstrate your understanding of the different constructs covered in the course, and your ability to use them in real code. There are three suggested themes, explained in further detail below. You must choose one of these:

- An investment trading app.
- An adventure game.
- A business software dashboard.

This document describes:

- The **requirements** your project must satisfy.
- How to **submit** your project for assessment.
- How to get **feedback**.
- The proposed project **themes**.

Requirements

Miniprojects will have to satisfy certain requirements in order to obtain marks:

- **Basic requirements** of presentation, delivery and design. You may get no marks if you ignore these.
- **Three levels** of accomplishment, each requiring the use of more sophisticated programming concepts.
- You will get **extra** marks for additional features of your choice, going beyond course contents.

Basic requirements

At all levels, you will be required to follow consistent and clean style guidelines, and to write comments. See the following link for reference: `https://www.cs.cornell.edu/courses/JavaAndDS/JavaStyle.html`. You must follow the naming conventions. Other style choices will not be enforced, but you will have to be consistent in your project. Write tidy code that is easy to read.

Your code must compile using `javac` and run with `java`, with the version installed on the ITL machines, and works as you have specified in the **Miniproject form**.

Programs will also be required to be reasonably well designed. While this may be to an extent subjective, there are some clear indicators of bad design which will not be allowed, such as:

- Your whole program is in one big file.
- There is a huge class with hundreds of lines of code, and/or tens of methods.
- There is an excessively long method, with hundreds of lines.

Levels of accomplishment

In order to get marks for each level, your program should correctly implement the properties on the right-hand side of the table below. Shortcomings with respect to these properties will result in deductions. Note that these levels are progressive. Before you can design a good inheritance hierarchy, you need a preliminary idea of the classes to be implemented. Before you can think of a GUI, you sould have a solid underlying object-oriented foundation.

Level 1 (10 marks)	• Standard I/O (keyboard input and printing to terminal). • Good design, making sensible use of classes and objects.
Level 2 (10 marks)	• Sensible use of inheritance, making use of Java's features such as abstract classes and interfaces where necessary. • Sensible use of polymorphism.
Level 3 (10 marks)	• Use of ArrayLists or other classes from Java's Collection Framework. • Adequate exception handling. • Graphical User Interface. • File I/O for features requiring permanent storage.
Extra (5 marks)	Add something special. Do your own research and use programming concepts that go beyond what you have seen in class.

Submission

In order to submit your project, you must use the corresponding activity on QM+. You must submit a zip file containing the source code and the filled-out **Miniproject form**. The deadline will be published soon, but it will be before the end of week 12.

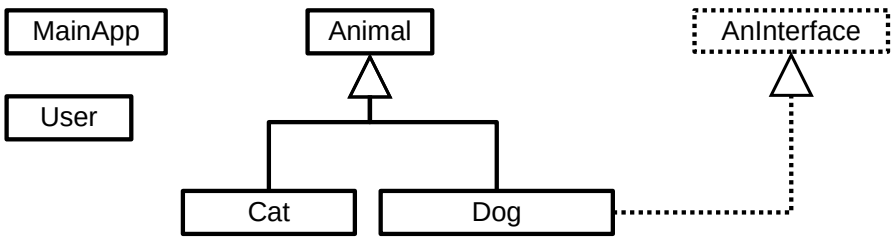
Class diagram format

In the Miniproject form you will have to include a class diagram in the following format (simplified UML):

- Each class should be represented as a box containing the name of the class.
- Each interface should be represented as a dotted-line box containing the name of the interface.
- Inheritance should be represented as an arrow (black line, white fill) from subclass to superclass.
- Interface implementation should be represented as a dotted-line arrow (black line, white fill) from class to interface.

Example:

```
class MainApp{}

class User{}

interface AnInterface{}

class Animal{}

class Dog extends Animal implements AnInterface{}

class Cat extends Animal{}
```



Feedback

It is important that you start working right away on your project, and get feedback often. You will be able to seek feedback through the following two channels:

- During the lab sessions from demonstrators and Lecturers.
- During office hours from Lecturers.

When you ask for feedback during the labs, please be specific, so that we have time for all of you. If you have coding problems, make sure you have made an effort to understand and solve them yourself before asking for help.

Themes

Here, more detailed descriptions of each theme are given.

Keep in mind that programming proficiency is the most important aspect of your project. A proper inheritance hierarchy is more important than the colour scheme of the UI, so don't lose sight of your priorities.

The increasing sophistication of the program should lead naturally to the use of the OOP constructs required at each level. For instance, in the investment trading app, at level 1 you might only implement buying and selling shares. Such a simple app may not require a complex inheritance hierarchy. At level 2, however, you might want to add different types of financial products, which justifies the use of e.g. polymorphism.

Be creative. These descriptions are just high-level guidelines. You have freedom to implement the program the way you like.

Investment trading app

This program should simulate a stock market. The user should be able to buy and sell securities, view their portfolio, deposit and widthdraw funds etc.

At increasing levels of accomplishment you can implement different types of financial products (e.g. options and cryptocurrencies), user profiles, graphical plots of price series, etc.

Adventure game

The title is self-explanatory. The game can be text-based.

At increasing levels of accomplishment you can implement different types of characters (player or non-player) and items, save/load features, a fancy GUI, levels of difficulty, etc.

Please keep in mind that creating graphics can be time-consuming, and your project will be assessed based mainly on programming proficiency.

Business software dashboard

This program should implement a dashboard to manage and monitor a business. The program should display Key Performance Indicators (KPIs) (e.g. sales, production figures, marketing reach metrics, human resources...) and allow basic operations (purchases, hiring...).

At increasing levels of accomplishment you can implement more KPIs, more types of operations, a staff hierarchy, complex production structures, etc.