ECS414U Miniproject Booklet
January 2021

Learning Outcomes

On passing the module you will be able to:

- tackle more complicated programming problems than in the first semester
- decompose programming problems using principles of object orientation
- use most of the object-oriented programming concepts when writing programs
- test and debug simple programs to ensure they work correctly
- explain programming constructs and argue issues related to programming

Programming mini-project

You must write a mini-project program. It should be chosen to demonstrate your understanding of and ability to use the different constructs covered in the course.
You should choose one of these projects. You may wish to keep notes on your progress week by week in a log book (for example including a copy of the current program listing and a description of what it does and how it works, or what does not work).

Mini-projects are divided in 3 levels.

You are expected to develop your program in stages – modifying your earlier version (that has been marked and achieved a given level of proficiency) for the next level version. Once your program has reached a level (see below) you should get it marked. If you are confident then you may wish to skip getting some levels marked. However, all mini-projects must be marked **at least two times**, and the level achieved confirmed in writing. For each level you will be required to state what your program does, the grade that you believe the program should obtain and also explain why it deserves that grade. As with the short programs you may be asked questions on how it works etc – the mark is for you convincing the tutor that you have met the learning outcomes not just for presenting a correct program.

In the following pages are example programs with indicative grades for different levels of development. However, your program does not have to do exactly as outlined in the descriptions for each level, these are just guidelines to help you through the task: use your imagination (though obey specific restrictions)! The more original is your project the better it will be.
What matters is that you demonstrate you can use the different object oriented concepts (objects, inheritance, polymorphism, etc.) and programming constructs like ArrayLists, file I/O, GUIs, etc., have achieved all the other criteria for a given level as described rather than the precise thing you have achieved. All students' programs should be different in what they do.
What matters is how many criteria you have achieved in your program.
Please avoid to cooperate with other students on the mini project. The more we see similarity in projects the more suspicious that will look the lower the marks awarded...

If you want to do a mini-project not suggested in this booklet then you need to talk to the module leader and have him to agree to the topic. If it is not agreed, it will not be accepted. Whatever the topic you should sketch a 3 levels scale of achievement similar to the ones for the suggested topics. So if you think of a mini-project topic non suggested in the following pages you should write down a 3 levels scale of achievement similar to the ones for the suggested topics to show to the module leader for approval. Only students with some previous Object Oriented programming experience are encouraged to original suggestions.

Suggestion 1: A stock market investor simulation

Write a program that simulates an investor buying and selling shares in a stock market. The investor needs tools to display his portfolio and buy and sell shares. Shares value can go up and down so un underlying simulation of the market should be implemented

—————————————— Level 1 –

Includes **screen output** and **keyboard input** and **basic classes**.
There are java source files for at least three major classes in the program.
Good source comments and code indentation is expected for all implemented parts of the code
Example: The program reads and prints the names of shares and their basic characteristics.

Includes methods and variables for at least five major classes, and all constructions above.
At least 3 major methods fully implemented and working for each class
Example: As above, but also the notion of the buying and sell shares is shown.

At least five major program classes will be implemented, with methods working and well designed, and all constructions above
Use of **inheritance** with at least one superclass and three subclasses
Each of the subclasses will add at least one useful and sensible property, and each of the subclasses will override at least one method from the superclass in a sensible and useful way (eg. by refining a calculation done by a superclass method). toString methods (or similar) where the subclass method returns a description of itself are not on their own sufficient to meet this requirement. The additional properties and overridden methods must be used in the code.
Class, method and variable naming will be clear and consistent
Example: As above, but also there is a basic simulation of the stock market, though most details, commentary etc. may be very simple

—————————————— Level 2 –

Polymorphism should be used in at least three subclasses, and all constructions above
At least four major program classes will be implemented, with methods working and well designed.
Code uses **substitution principle** and **dynamic binding.**
Comments are clear and applied to class and method level consistently
Example: As above, but the simulation is more natural, there is a running commentary of the change in value of the shares with major events reported.

—————————————— Level 3 –

Use of **ArrayLists or other classes from Java's Collection Framework** in all parts of the program, and all constructions above.
Exception handling is carried out appropriately in all parts of the program
Inheritance is correctly applied to all parts of the program.
Example: As above, but all types of accounts and functionality will now be included in the simulation. The simulation is now mostly ruled by a basic **GUI**.

Includes **file input and/or output**, and all constructions above
The simulation (including player movement) will be displayed on the GUI
Polymorphism will be fully implemented in all parts of the program
Example: As above, with a full GUI now controlling all aspects of the simulation; data should be read from files.

For top marks add **something special** (using other more advanced constructs or algorithms, or something you just read up on yourself). Make it a program someone would really want to use!
Example: .. real historical data, moving averages,...

Suggestion 2: An adventure game

Write a program that implements a simple (yet exciting!!!) adventure game. Different groups of characters (e.g. magicians, warriors, monsters, different types of warriors, magicians, etc.) will have different characteristics, strengths and behaviour. Heroes may pick up objects on their way, can engage in fights when in close proximity to each other, etc. At least 6 characters from 3 different groups, and 2 different types of object should be included in the game.

———————————— Level 1 –

Includes **screen output** and **keyboard input** and **basic classes**.
There are java source files for at least three major classes in the program.
Good source comments and code indentation is expected for all implemented parts of the code
Example: The program reads and prints the names of the characters and their basic characteristics.

Includes methods and variables for at least five major classes, and all constructions above.
At least 3 major methods fully implemented and working for each class
Example: As above, but also characters, objects, etc. are arranged on a simple layout.

At least five major program classes will be implemented, with methods working and well designed, and all constructions above
Use of **inheritance** with at least one superclass and three subclasses.
Each of the subclasses will add at least one useful and sensible property, and each of the subclasses will override at least one method from the superclass in a sensible and useful way (eg. by refining a calculation done by a superclass method). toString methods (or similar) where the subclass method returns a description of itself are not on their own sufficient to meet this requirement. The additional properties and overridden methods must be used in the code.
Class, method and variable naming will be clear and consistent
Example: As above, but also there is a basic simulation of the game, though character movement and interaction may be very simple.

———————————— Level 2 –

Polymorphism should be used in at least three subclasses, and all constructions above
At least four major program classes will be implemented, with methods working and well designed.
Code uses **substitution principle** and **dynamic binding.**
Comments are clear and applied to class and method level consistently
Example: As above, but game simulation is more natural, characters can move a specified distance for each turn.

———————————— Level 3–

Use of **ArrayLists** in all parts of the program, and all constructions above.
Exception handling is carried out appropriately in all parts of the program
Inheritance is correctly applied to all parts of the program.
Example: As above, but also basic character interaction scenarios will be included (e.g. fights). A basic **GUI** will also be included showing character details, status, etc.

Includes **file input and/or output**, and all constructions above
The simulation (including movement) will be displayed on the GUI
Polymorphism will be fully implemented in all parts of the program
Example: As above, with full character interaction and with full interaction with objects in the game (characters pick up objects etc.). Characters' details should be read from a file.

For top marks add **something special** (using other more advanced constructs or algorithms, or something you just read up on yourself). Make it a program someone would really want to use!
Example: Use a smart algorithm to determine the results of fights, use a fancy GUI for the game. Or ...

Suggestion 3: A car racing simulation

Write a program that simulates a car racing. You should include different types of cars (e.g. by brand or engine size or horsepower, etc) with different abilities (e.g. braking, straight line speed, acceleration), characteristics (e.g. aerodynamics, poor at cornering, tends to overheat) etc. Notice the project can be very difficult if you try to implement "arcade racing" where you see the race through the windscreen (you need 3D for that); a simpler simulation is looking at the race from above (you just need 2D for this)

——————————— Level 1 –

Includes **screen output** and **keyboard input** and **basic classes**.
There are java source files for at least three major classes in the program.
Good source comments and code indentation is expected for all implemented parts of the code
Example: The program reads and prints the names of cars and their basic characteristics.

Includes **methods** and **variables** for at least five major classes, and all constructions above.
At least 3 major methods fully implemented and working for each class
Example: As above, but also the structure of the track is shown.

At least five major program classes will be implemented, with methods working and well designed, and all constructions above
Use of **inheritance** with at least one superclass and three subclasses.
Each of the subclasses will add at least one useful and sensible property, and each of the subclasses will override at least one method from the superclass in a sensible and useful way (eg. by refining a calculation done by a superclass method). toString methods (or similar) where the subclass method returns a description of itself are not on their own sufficient to meet this requirement. The additional properties and overridden methods must be used in the code.
Class, method and variable naming will be clear and consistent
Example: As above, but also there is a basic simulation of the race, though most details, commentary etc. may be very simple

——————————— Level 2 –

Polymorphism should be used in at least three subclasses, and all constructions above
At least four major program classes will be implemented, with methods working and well designed.
Code uses **substitution principle** and **dynamic binding.**
Comments are clear and applied to class and method level consistently
Example: As above, but race simulation is more natural, there is a running commentary of the race with major events reported.

——————————— Level 3–

Use of **ArrayLists or other classes from Java's Collection Framework** in all parts of the program, and all constructions above.
Exception handling is carried out appropriately in all parts of the program
Inheritance is correctly applied to all parts of the program.
Example: As above, but all types of cars and functionality will now be included in the simulation. The simulation is now mostly ruled by a basic **GUI**.

Includes **file input and/or output**, and all constructions above
The simulation (including player movement) will be displayed on the GUI
Polymorphism will be fully implemented in all parts of the program
Example: As above, with a full GUI now controlling all aspects of the simulation; data should be read from files.
For top marks add **something special** (using other more advanced constructs or algorithms, or something you just read up on yourself). Make it a program someone would really want to use!
Example: realistic animation, sound etc

Suggestion 4: A business software dashboard simulation

"Dashboard provides at-a-glance views of key performance indicators (KPIs) relevant to a particular objective or business process (e.g. sales, marketing, human resources, or production)" (from wikipedia). You should be familiar with basic business concepts for this project. You should have an underlying simulation of real world events that determine the evolution of the business in terms of materials acquired, products sold, hiring, forecasts etc...

———————————— Level 1 –
Includes **screen output** and **keyboard input** and **basic classes**.
There are java source files for at least three major classes in the program.
Good source comments and code indentation is expected for all implemented parts of the code
Example: The program reads and prints the names of basic dashboard items and their basic characteristics.

Includes methods and variables for at least five major classes, and all constructions above.
At least 3 major methods fully implemented and working for each class.
Example: As above, but also the notion of the hiring, firing people, buying, selling items, forecasting is shown.

At least five major program classes will be implemented, with **methods working and well designed**, and all constructions above
Use of **inheritance** with at least one superclass and three subclasses.
Each of the subclasses will add at least one useful and sensible property, and each of the subclasses will override at least one method from the superclass in a sensible and useful way (eg. by refining a calculation done by a superclass method). toString methods (or similar) where the subclass method returns a description of itself are not on their own sufficient to meet this requirement. The additional properties and overridden methods must be used in the code.
Class, method and variable naming will be clear and consistent
Example: As above, but also there is a basic simulation of the system, though most details, commentary etc. may be very simple

———————————— Level 2 –
Polymorphism should be used in at least three subclasses, and all constructions above
At least four major program classes will be implemented, with methods working and well designed.
Code uses **substitution principle** and **dynamic binding.**
Comments are clear and applied to class and method level consistently
Example: As above, but the simulation is more natural, there is a running commentary of the system with major events reported.

———————————— Level 3–

Use of **ArrayLists or other classes from Java's Collection Framework** in all parts of the program, and all constructions above.
Exception handling is carried out appropriately in all parts of the program
Inheritance is correctly applied to all parts of the program.
Example: As above, but all items will now be included in the simulation. The simulation is now mostly ruled by a basic **GUI**.

Includes **file input and/or output**, and all constructions above
The simulation (including player movement) will be displayed on the GUI
Polymorphism will be fully implemented in all parts of the program
Example: As above, with a full GUI now controlling all aspects of the simulation, displaying different dashboard elements; data should be read from files.
For top marks add **something special** (using other more advanced constructs or algorithms, or something you just read up on yourself). Make it a program someone would really want to use!
Example: real data simulation and business transactions