ISYS 1118 Software Engineering Fundamentals

Assignment Draft **2019 Semester 1**

Objective

This SEF group assignment is assessed over 10 weeks through progress meetings (10 marks), two milestones (10 marks) and final face to face marking (20 marks). The progress meeting are designed to give early feedback where the tutor plays the role of the client as well as assessor. What is expected of you each week is clearly specified. The lab exercises over the first week are designed to make you familiar with collaboration, version-control, modelling and testing tools that facilitate a continuous integration (CI) approach. The first milestone nurtures an agile development approach placing more emphasis on analysing requirements, working software and test driven development (TDD). The second milestone requires refactoring the design and code, using UML notation, emphasising maintainability and extensibility. In the second milestone and the final face-to-face marking students will be required to present and justify their UML designs, how they are translated into code as well as demonstrating the final working software. Students are expected to work individually (50 hours) and as a team (30 hours) for this assignment in addition to the time spent during contact hours.

Please note this assignment carries 40% of the overall marks for SEF.

Detailed Marking Scheme and Timeline 2019 Semester 1

1. **Progress Marks (10)**
2. **2 Milestones (2 x 5)**
3. **Face-to-face marking (20 marks)**

1. Progress Marks (10 marks)

Wk	Suggested Activities (these may vary between teams but students are expected to show progress)	Progress	Milestone
3	Select the project and form the team – consider the strengths, interests and experiences of team members	1 mark	
4	Identify User stories for Product/ Release Backlog	1 mark	
5	Identify main domain classes and add responsibilities such that objects can interact to get the first release done. Distribute classes/responsibilities to members.	1 mark	
6	Write Test Cases and start writing code to make them pass (individual)	1 mark	
7	**Initial Milestone**: Present partially completed product and get feedback from tutor and class. Demonstrate working Test cases (individual) and continue integrating code and testing.	1 mark	5 marks (see section 2)
8	Plan second release incorporating feedback. Add classes, extend class responsibilities and distribute work.	1 mark	
9	Develop detailed Group UML Diagrams (class, use-case).	1 mark	
10	Develop Individual UML Diagrams (sequence, state, object)	1 mark	

	Demonstrate individual Test cases		
11	Develop Activity Diagrams. Combine code and run integrations tests. Resolve conflicts. Refactor design and code before final submission.	1 mark	
12	**Second Milestone**: Present final product to tutor and class.	1 mark	5 marks (see section 2)
13	**Face to Face Marking of design and code by Class Tutor**		20 marks (see section 3)

2. Milestones 1 and 2

Milestone 1 (Total 5 marks) in Week 7

This first demo/presentation will award marks for both group and individual work. Each individual member is expected to undertake at least two major tasks or responsibilities. Some of these responsibilities can be shared. For example, in the chess-like game "show all valid move destinations for a piece" can be a user-story/responsibility that must be shared by the Piece and the Board classes. The subclasses of Piece such as Rook and Knight may know valid moves but only the Board classes knows where the vacant and opponent squares are located. In such a situation, those members writing the Rook and Knight classes may override the method getAllValidMoves():Squares[], while the person developing the Board class may write a method to check which of these squares are free to move using IsValidDestination(square). These two methods from different classes may thus be combined to determine all valid destinations. Students can defer non-model (GUI) aspects until the "core-logic" is fully tested (by using a simple textual interface).

Activity	**Marks**
Group Part	
Presenting the Initial Release Backlog (design documents: Class and Use Case diagrams) and demonstrating a partially completed product	2 marks (4 mins)
Individual Part	
1. Individual member is expected explain his or her role/obligation towards carrying out the main user-stories/responsibilities, and any challenges faced. 2. Individual member is expected to demonstrate how these obligations are discharged, by running some test cases. If code implementing these tests is not complete, present the test cases designed. If little or no progress has been made briefly explain the reasons.	3 marks (4 x 1.5 mins)

Milestone 2 (Total 5 marks) in Week 12

The second demo/presentation will award marks for both group and individual work though more marks is awarded for group work.

Activity	**Marks**
Group Part	
Presenting the Final Release Backlog and demonstrating the final product	2 marks (3 mins)
Use a class diagram to present the overall design; use two sequence-diagrams to present the core functionality. Explain the design rationale highlighting usability, maintainability, extensibility etc.	2 marks (3 mins)
Individual Part	
Individual members explaining their role/ contribution and any challenges	1 marks (4 x 1 min)

3. Face to Face Tutor Marking (Total 20 marks)

This final assessment point will award marks for both group and individual work with equal emphasis on individual and group contribution to design and coding. The group effort will be measured through overall class and use-case diagrams, which should explain the overall requirements and design decisions. Each individual member is expected to demonstrate their individual contribution by detailing their part of the use-case (a use case textual description) and class-diagrams and by coming up with two other UML diagrams (a sequence diagram and **one** of state/activity/object diagrams). **Tutors will check your diagrams are aligned with your code and how group diagrams such as class and use case diagrams are aligned with individual diagrams (sequence, state). Include a readme file identifying percentage contribution of each member allowing the group mark to be adjusted accordingly.**

Activity	**Marks**
Group Part	**Total 10 marks**
Overall Class Diagram (group) The class diagram must show all the methods, attributes, associations and multiplicities. Name the associations wherever possible. Up to 2 marks for valid class diagram that captures all the main classes, associations (with naming) with the correct multiplicities. You are expected to follow the standard UML notation. The remaining 1 mark is allocated for consistency with individual sequence diagrams (you may have to update the class diagram to reflect the final sequence diagrams).	3 marks
Overall Use Case Diagram (group) Up to 2 marks for a valid Use case diagram capturing all individual use cases. For the remaining 1 mark show all the relationships including extends/includes.	3 marks
Group Evaluation (max 1 page report) including: Lessons learnt about project/process management (2 marks) Strengths and weaknesses of your system in terms of reusability, extensibility, maintainability and usability (2 marks). 2+2 = 4 marks	4 marks
Individual Part	**Total 10 marks**
Show detailed methods/attributes/associations relating to individual contribution in the class diagram	2 marks
Show detailed explanation relating to individual contribution in use case diagrams – present one use case textual description for a significant use case.	2 marks
Sequence Diagram (individual)	2 marks
One of state/activity/object diagram	2 marks
Code explanation relating to individual contribution. The code is expected to reflect the design. Code quality and explanation (code must be refactored)	2 marks

Submission Details: 31st May 2019 4.30 pm. No late submission allowed.

All source code and diagrams must be printed and bound together and be submitted to the School of engineering office (10.09.03) by 4.30 pm on the last day of the semester (31st May 2019). You must also upload a zip file called FinalSubmission.zip to your groups Trello board.

FAQ

1. Can I use other OO languages (e.g. Groovy and Ruby) and technologies (e.g. Angular –JS and Node-JS) for implementation? You must seek permission with your tutor. While we are happy to encourage you to learn new technologies, there may be no additional support from the teaching team. Moreover, the UML diagrams and code required for the second milestone assumes an OO design/implementation is followed.
2. Do I need to write the test code in JUnit? Yes, unless you are using another OO language such as Groovy, Ruby or Android which have their own API. JUnit is a standard API that comes as part of Eclipse. JUnit Information available on: http://junit.org/faq.html. JUnit will also be covered in the lecture/tutes.
3. Can I use one of the free Cloud based platforms to host my system? We are happy for you to explore this option but we cannot provide any additional support.
4. Will there be a client for my software? Yes, your tutor will play that role, who will also assess both demos/presentations and final design/code. In addition there will be a discussion board for assignments. One of the team members can also take the role of product owner.
5. What additional help can I get? We will arrange additional student consultation times if necessary. The lecturer will announce specific times on the blackboard.
6. Do I need to show exceptions in class diagram? No they will clutter the diagrams.
7. What additional Java features (not taught in introductory programming courses) will be needed or useful? You may want to use Swing, JCF, Abstract classes, interfaces, exceptions etc., which can be found in any Introductory Java Programming books (such as that by Daniel Liang). Those who are developing distributed or web-based applications may consider using sockets, threads, servlets and Java server Faces/ Java server pages which can also be found in most introductory programming books (such as the book by Daniel Liang).
8. Why are we encouraged to use a MVC-based-architecture? It allows you to focus on the model initially (data structures and algorithms) without having to worry about the view aspects. Loose coupling in MVC also makes the product more maintainable.
9. What else can I do to make my product more extensible and maintainable? Consider using common design principles and patterns.
10. Why does this course require programming? Students find it difficult to know whether their UML design is good or even correct, without implementing the design. Moreover, implementing the design and measuring the tangible outcomes promotes an iterative approach resulting in improved designs. Implementing your design and testing whether the product meets the original requirements allows you to go through the whole software engineering lifecycle, thus preparing you better for your final year projects and your working career in the software industry. Note, student-mentoring support is provided for SEF programming related issues from week 3.
11. Why insist on teamwork? Teamwork and communication are the two main attributes employers are looking for in IT/SE graduates – and this project provides a great opportunity. Please be honest and open about your strengths and weaknesses, and seek help from other team members, teaching staff and mentors as early as possible.

Assignments: Summary, Rationale and Common Aspects

Students with differing backgrounds and programming experience currently enrol into SEF, as it is a core course for programs (CS, SE, IT). To cater for such diversity, students are allowed to choose one of the projects from the given list, matching their background and interest. These assignments are designed to facilitate an agile or test driven approach for the first milestone, and a more rigorous design and implementation for the second milestone using UML notation and design/code refactoring. We have tried hard to come up with interesting and varied projects to match your interests and backgrounds. Your team should consider the strengths and interests of your members when selecting the project. You may also want to do some reading, testing and feasibility studies before finalizing your assignment selection in week 3. The first milestone is designed to enable students to develop some working-software or a prototype as soon as possible and get some feedback during the class presentation in week 7. The second milestone will focus more on the design aspects including extensibility and maintainability of the software.

In all of these assignments you are required to show weekly progress during your labs and make use of tools used in the industry such as Trello (Collaboration), JUnit (Testing), GitHub (Version Control) and Lucid Chart (UML Design). Guided lab exercises will help you familiarize with these tools. All students are expected to have some Java Programming experience through previous or current course offerings. The book, Introduction to Java by Daniel Liang will cover all the Java knowledge required for this course. In the initial part, students with limited programming skills may encounter some steep learning curve; but our experience shows project based experience in SEF help improve student grades in other programming courses.

Most assignments include some suggestions for initial test cases in addition to core requirements; please feel free to replace these test cases with others and come up with additional requirements that will enhance the functionality of the system. In the initial weeks each team is expected to spend some time coming up with specific classes and methods, which can collaborate to meet all the requirements. The features for bonus marks are to be attempted only after completing all the required tasks.

Assignment	Description
Student Advisory System	Presents a familiar domain dealing with student enrolment and program maps. GUI is optional.
Budding Share Market Investor	Ideal for those who have some background in finance and interest in distributed computing.
Simple Strategic Game	Pacman like game should make for an interesting and enjoyable game. Requires some knowledge of GUI and events. Some start up code may be provided/discussed in the lectures.
Supermarket Support System	Presents students with a familiar domain model involving domain classes Product, Sale, Customer etc. GUI is optional.
Traffic Light Simulator	This project will be an ideal one for those interested in flexible design patterns and graphical simulation. Requires some knowledge of GUI, patterns and use of events.
Chess like Game	This strategic game based project is much simpler than the traditional chess but allows students to develop their OO design and programming skills. The second milestone requires the use of Swing/FX classes.
Vacation House Exchange Club	This project is suitable for those interested in using a web-based framework or a client-server architecture (RMI). Common frameworks include Groovy on Grails, Play, Rails, Stripes, JEE or Spring framework; however there will be no technical help in these frameworks that may involve some steep learning curves. You may also consider using multiple technologies. Suitable for independent learners or those with some prior working experience.
Monster Game	This fun multiplayer project is suitable for those interested in working with distributed applications. It will require the use of sockets, threads and synchronization or RMI. This is probably one of the most technically challenging assignments. If selecting this project you need very good programming and debugging and design skills.
Task and Staff Scheduler	This project will be great for those considering a project management career; it will give you a head start with scheduling and resource allocation, a common problem in project management. This projects requires some use of GUI and events.
Collaborative Snakes and Ladders Game	This project requires you to extend the Snakes and Ladders game for which some sample code is provided.

The eXcel-lent Commuter project	The client wants an online tool that helps the eXcel London conference visitor to use the London tube and Docklands Light Railway(DLR) network to get to the venue.
The Hyde Park/Kensington Gardens Alert project	Your client, representing security for HRH Queen Elizabeth II, wants a solution that can effectively provide snapshot of royal and government personnel available who can respond to a number of major incidents in this popular park.
The RA Summer Exhibition project	Client wants to improve the experience of visitors to Royal Academy and stop printing booklets for next year.
The Conference Tracker project	Conference staff, exhibitors and general visitors to the this year's TECHXLR8 conference at eXcel London are issued barcoded conference ID tags to recorded at entry after security checks. The client would like to make more use of this ID to collect data on different conference activities such as visitor engagement at exhibition booths and stalls, attendance at a lecture or workshop, or even purchases at food and beverage locations.
Progress of Children at Kinder	Client want a system to keep an eye on their children's progress at Kinder.

Note: Each group member must be from the same tutelab session.

Project 1: Student Advisory System

You are required to develop a stand-alone student advisory system to help CSIT students select the most appropriate course offerings for the current and future semesters allowing them to complete the degree requirements at the earliest. Assume there can be more than one program (degree), and each program will consist of a number of core courses, school electives and free choice electives. Both core courses and school electives can have one or more prerequisites. Free choice electives do not have any prerequisites and the names of these electives need not be captured (just state them as free). Assume also courses may be 12, 24 or 36 credit points and total credit points for course offerings in any one semester cannot exceed 48. Any student failing a course three times should be excluded. Students are mandated to follow the advice though it can help to inform them of the appropriate choices.

Your system should allow various actors to interact with the system.

- School admin to specify the set of core courses, the set of school electives, and their prerequisites for each program (CS, IT, SE). Assume a course can have only one prerequisite and that the prerequisite can be stated as one from a group. For example, students can do Further Programming after completing either Programming Techniques or Programming 1. Note a course that is a core for one program need not be a core for another.
- Program manager to add course offerings for the current semester and the next seven semesters. Note the future course offerings may be subject to change reflecting staff and space constraints. The program manager is also capture any overlap in course offerings to avoid students enrolling into two courses offered at the same time.
- The course coordinator may waive the prerequisite for a specific student and it should be captured in the system.
- Students should be allowed enter all the courses passed, failed or exempted since commencement into the program including school and free electives. Students should be allowed to rank the school electives offered in the current semester in order of preference.
- Students should be able to give the option to view the courses they can enrol as well the best combination of courses (individual program map) allowing them graduate at the earliest. The free electives need not be named.

Out of Scope

- No advice should be provided for excluded students (failed a core three or more times) and should be directed to see the program manager to make special arrangements.
- Currently only a stand-alone system is needed and no concurrent access are permitted.
- There may be many different strategies and heuristic algorithms suited for selecting courses but you are required choose one that allows them to complete the degree at the earliest time considering when these courses are offered in the current and future semester.

Additional Requirements

- All the data entered to be stored (in files, through serialization or database access).
- You should provide a facility for all actors to edit the current data. For example, students should be able to add other course passed, failed or exempted while the program manager may update the courses offered this semester or the overlaps between them (in case the lecture venues are changed).
- Create some dummy data for 2 different programs, course offerings for the current and future semesters and the data for 5 students made up of all the results and exemptions granted in the past. Create separate test cases to demonstrate all the main functionalities. The data for the programs should be based on the description of two of the three programs (CS, IT and SE)
- You are allowed to make any reasonable assumption but please state these explicitly in your documentation.

Bonus Features:

- Securing data access to various roles and individuals.
- User friendly Interface
- Demonstrating extensible, maintainable design

Possible Domain Classes

- Program, Course, CourseOffering, Elective, Core

Some Suggested Test Cases for Milestone 1 & 2

1. Test that a student cannot enrol in a course that has no offering in the current semester
2. Test that a student cannot enrol in a course without the necessary prerequisite.
3. Test that a student granted exemptions is not required to enrol into them.
4. Test students are not advised to enrol into two current overlapping course offerings.
5. Test the program map suggested has no two identical courses.
6. Test the courses selected for a given semester do not add up to more than 48 credit points.
7. Test the system selects courses allowing them to graduate at the earliest.
8. Test the system respects student preference for school electives.
9. Test admin, program manager, course coordinator and student functionalities can be accessed only with valid username and password
10. Test the system is able store and retrieve the data.

Project 2: Budding Share Market Investor

You are required to write a stock market game to introduce budding investors to learn the risks and opportunities share market-trading presents. Users should be given $1,000,000 (bogus money) initially and be allowed to buy and sell the shares at current price from the ASX. Assume the brokerage cost is made up of two parts: a fixed charge ($50) and a charge that is a percentage of sale or purchase (0.25% for sale and 1.0% for purchase) that may be lowered when high volumes are traded. **Please limit to 6 ASX shares** of your choice.

Users should be provided all of the following functionality:

- Register as a stock market player
- Login as a user
- Open a Trading Account
- List the average price of shares in possession and current number of shares held
- Queue to buy or sell specified number of shares. The transaction should take place if the current share price is lower than or equal to the buy price queued or if the current price is more than or equal to the sell price queued. You may assume that the specified number of shares can be traded at the queued price (not in real life).
- Reject purchases or sales exceeding $600,000.
- Remove the shares queued for buying or selling (before transaction is done)
- Track the movement of share price (at regular 1 hour intervals) and plot the graph
- Specify cut loss options in case of a stock market crash
- List a summary of transactions within specified dates
- View current balance in dollars and current stock value
- Update the balance of top budding traders at the end of each trading week

Admin should be able to:

- Delist a member.
- List players trading amounts exceeding $100,000 within a week.
- Lower the brokerage cost of specific players
- Modify parameters (e.g. trade limits and frequency of share price updates)

Bonus Features

- Encode specific strategies for automatically adding or removing items to buyer and seller queues. For example, if the overall stock market index drops by 1% within an hour all existing shares may be put for sale at current stock prices. You may come up with a number of strategy classes to suit the credit-risk level of investors.
- Develop a separate server that simulates the stock movement to test the behaviour of your strategies. In this case Share market application will connect to this server instead of SGX.

Some Considerations

This project will be of above average difficulty. You may have to work with (delayed) stock prices (unless you are willing to pay). You must have some interest and knowledge about web services, XML, distributed computing etc. If using Java, you may consider using servlets, RMI, web services or sockets.

Possible Domain Classes

Customer, TradingAccount, Transaction, QueuedItem, Admin etc.

<u>Suggested Initial Test Cases for Milestone 1</u>

1. Trading can start with valid (registered) user ID and password
2. Trading cannot start without valid user ID and password
3. Test purchasing specified number of share results in customer stock level for the share being increased by the specified number
4. Test whether the brokerage price is computed correctly for a sale
5. Test whether the brokerage price is computed correctly for a purchase
6. Test whether average share price for existing shared held by the customer is computed correctly based on three or more purchases
7. Test whether an item queued can be removed successfully before transaction is done
8. Test that an item cannot be removed from the queue after the transaction is done
9. Test attempts to queue a purchase or sale for amounts exceeding $600,000 are automatically rejected.

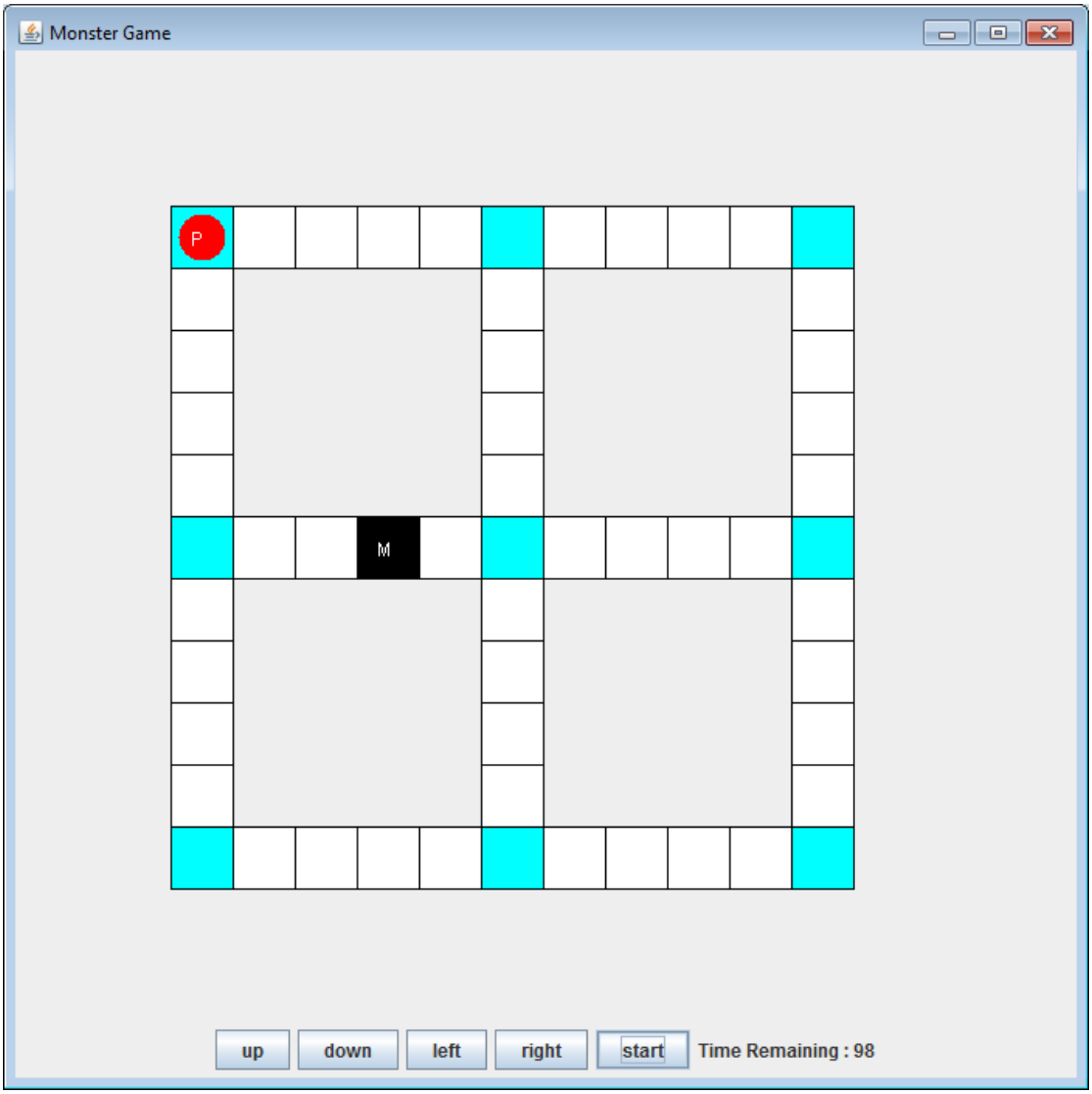
Project 3 :Single Player Pacman like Game

1. Objective

This assignment uses an interactive game as a vehicle to allow you to learn and demonstrate your analysis, design, implementation, teamwork and testing skills. This assignment will require a fair amount of object oriented design and programming skills. You will be provided with some sample code using Swing classes for GUI but you need not follow the design or coding.

2. Overview

You are required to develop a stand-alone Monster-Chase-Player-Game using Java. The main objective of the player is to stay alive by outrunning, escaping, jumping over, setting traps or by placing roadblocks. The marauding monster(s) too may have many tricks such as mounting invisible attacks, using better chase strategies or by giving birth to more monsters. The users should be allowed to combine any player type with any monster type to have a different gaming experience. You may change the game rules specified below (if it makes the game more playable) but it should allow different player and monster capabilities to be specified at runtime.



3. Main Assignment Tasks and Expected Learning Outcomes

This section briefly highlights how this assignment meets course specific learning/teaching objectives.

I. User Input

Handle mouse and/or keyboard events:

- Use button or key events to control player movements
- Other appropriate buttons to start, pause, restart.

II. Varying Player/Monster Capabilities (Algorithms, Specialization)

Allow different capabilities for Monster and Player objects:

- Derive classes for specific capabilities
- Use existing behaviour and state information from base classes

III. Allow different Monster/ Player Capabilities at run-time (Dynamic Binding)

Use of appropriate OO techniques that promote dynamic binding:

- Allow users to choose required behaviours/capabilities at runtime
- Use of polymorphism to facilitate dynamic binding
- Use of abstract/normal methods that can be overridden in subclasses to produce different behaviours

IV. Improved Game Experience (Usability, Playability)

To improve game experience:

- Allow different skill levels by varying game duration and update frequency (delay player monster updates using Thread.sleep())
- Allow layout to be changed reflecting user preferences
- Allow games to be paused

V. Tracking Registration Details and High Scores (Persistence)

Handle files and exceptions:

- Store and retrieve user records and score details in files
- Request users to login with valid username and password

VI. Group Project (Group work and Team working skills)

Team design, task delegation, unit testing and documentation:

- Develop detailed classes/methods and refactor as necessary
- Develop test harnesses to facilitate unit testing
- Develop good coding style/documentation (comments)

4. Detailed Assignment Requirements

You are presented with the skeleton code for a simple monster game using Java Swing classes. You are expected to extend these classes, add new classes/methods, provide additional functionality (such as registration, login, tracking scores) and finally refactor the design/code.

Player Extensions Required

1. Player needing energy to move

- Player needs 2 calories of energy to make each move.
- Eating (moving over) golden nougats found in each cell provide 6 calories.
- Player starts with 40 calories (can be varied)
- Player cannot move when it runs out of energy (just wait to be eaten-up or time-up)
- Player wins if player survives the game duration

2. Player needing energy with ability to speedup (multiple moves) (rules for 1 applies)

- Allow player to move multiple cells by pressing the button key repeatedly within the stipulated time slot.
- Maximum number of cells is limited to three.
- Each additional step in a move takes up double the energy (2 for 1 step move, 2+ 4 for 2 steps move, 2+4+8 for 3 steps move)

3. Player needing energy with ability to put traps (all rules for 1 apply)

- Allow player to put a trap in its current cell which prevents the monster from moving
- Putting a trap costs 50 calories and will last for 10 time units
- Assume traps are associated with cells

4. Player with limited manoeuvrability

- Player can only turn at 3 or 4 lane junctions.
- Player can place a roadblock up to 3 times.
- The roadblock takes 4 units of time for a monster to clear.
- The player can also escape to the centre of grid up to 3 times.

Monster Extensions Required

1. Hidden Monster

- Allow monster to be hidden until it is within 5 units of player.

2. Smart Monster with better chase strategy

- Reflecting the direction of player move.
- Reflect player manoeuvrability, energy level and the ability to speedup
- Consider the number of traps and their locations

3. Jumping Monster

- This monster can jump and eat any player which is in sight (horizontal and vertical)

4. Productive Monster (may require changing other classes)

- Gives birth to a new baby monster (use a different image) after stipulated time.
- Any baby monster will be automatically destroyed if any player walks over them in the first 10 time units.

5. Group Features Required

These additional features are related to usability and extensibility

- Allow users to specify player and monster types needed for the game
- Allow user to vary game parameters (such as the timeslot and the time duration)
- Allow game to be paused and restarted
- Allow user registration (name, address, username and password
- Allow user to login specifying username and password
- Track scores (wins, losses)
- Save and retrieve user details and score history
- Allow use of keys (instead of buttons) when appropriate

6. Overall Design and

The overall design is to be done jointly by all members. It is expected two members will implement the two each of the Player related functionality while the other two the two each of the Monster related ones. Utility classes (such as Cell and Grid) must be shared by all team members.

Bonus: (two of the following)

- Allow the grid to be changed at runtime
- Allow the game state to be saved and retrieved.
- Allow player to escape horizontally/vertically by connecting the left side to right side of the grid and topside to the bottom side of the grid.

7. Suggested Initial Test Cases for Milestone 1

1. Ensure the array keys can used to move along the grid.
2. Ensure the array keys cannot move outside the grid or inside the hollow region.
3. Check appropriate buttons/keystrokes for pause and restart
4. Verify that the game starts when a valid (registered) user ID and password combination is supplied
5. Verify that the game does not start when invalid user-ID and password combination is supplied
6. Verify the highest scores are maintained correctly.
7. Verify player specific functionalities using the standard monster
8. Verify the monster specific functionalities using the standard player.

Project 4: Business Intelligence System

A supermarket wants to introduce a business intelligence system allowing purchasing patterns of customers to be analysed. The swipe-card used by customers capture the address (post-code) and other relevant details of customers. To incentivise the use of the card, customers get 1 point for every $10 spent and are automatically discounted $5 for every 20 points. The system stores the product, customer, transaction and employee related details allowing necessary sales reports to be generated for business intelligence. Data may be stored in a relational database, flat files or through serialization. The system should store product, customer and employee related details allowing necessary reports to be generated.

Current Limitations

A typical checkout terminal consists of a touch screen and a barcode reader but for this assignment all data will be entered through the keyboard (including the customer card number and product ID). Assume the customer is required to specify the customer number thus allowing customer details to be stored.

Project Requirements

A typical customer:

- I want to be able to select the product name from a given list.
- I want to be able check the price of any item by keying in the ID before proceeding with the sale
- I want to check the discounts applicable for a specific product when purchases in bulk

The sales staff (Kim):

- I want to be able to login to the system the system and override the transaction details (removing item, cancellation) in case a customer has problems.

The warehouse staff (Frank):

I want to be able to replenish stock levels before placing items received on the shelves.

The manager (Tim): I want a system that:

- Maintains unit-price, stock level, replenish-level and reorder quantity for all items.

- Maintains supplier details for all products.
- Allows me to override the standard price for a specific product when there is a promotion
- Allows me to offer special discounts for bulk sales (for example, 10% discount for 5 items, 20% discount for 10 items etc.) on specified products
- Automatically place a purchase order for all items below replenishment level
- Generate a sales report for the specified period
- Generate a report on fast moving items based on value.
- Generate a report on sales based on customer address (postcode) to arrange for a marketing campaign.
- Generate supply report (Payments for supplies are out of scope)
- List products generating the most revenue.

Bonus marks (one of the following)

Extend into web-based or client server applications allowing concurrent access. This extension will require the use of sockets and/or RMI and/or Servlets access.

Interface to an external barcode reader

Possible Domain Classes

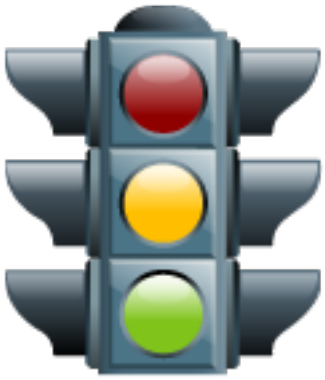
System**,** Sale**,** SalesLineItem**,** Product, Location**,** Customer, Credit- Card, Employee, Employee, Manager, SalesStaff, Supplier

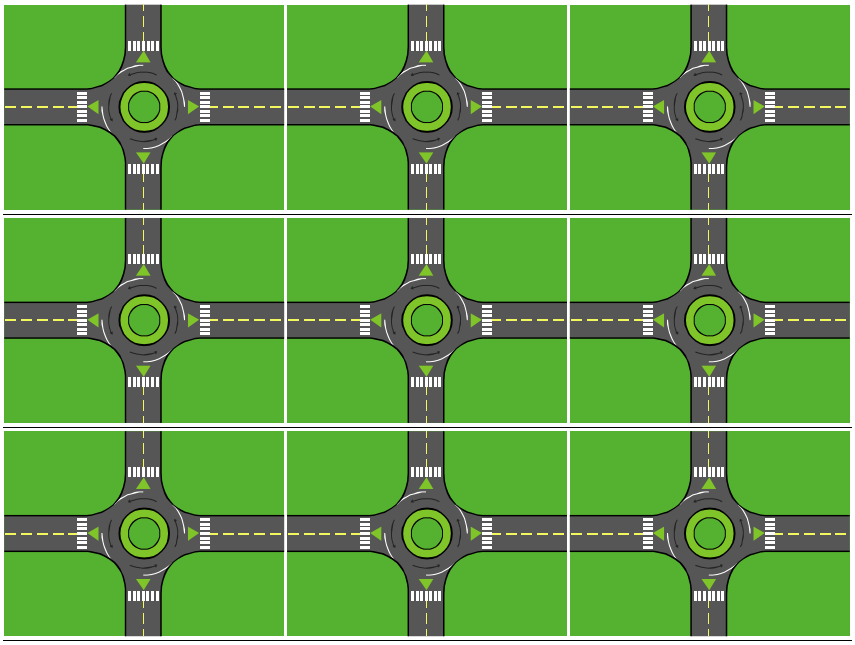
Suggested Initial Test Cases for Milestone 1

1. Test performing a sales reduces the stock level for all products in the sale
2. Test replenishing stock increases stock level by the specified amount
3. Test sale price is computed correctly based on sale line items
4. Test sale price is computed correctly for discounted items
5. Test sale price is computed correctly for items offering discounts for bulk sale
6. Test sale price is not affected for non-discounted items
7. Test the loyalty points are allocated correctly
8. Test Maximum discounts are automatically given based on current loyalty points at the end of transaction
9. Test discounts are computed correctly when loyalty points are combined with bulk discount
10. Test discounts are computed correctly when loyalty points are combined with some promotional items

Project 5: Traffic Signal Simulator

Design and implement a system, which will allow staff of traffic engineering department to simulate traffic signal timing for up to 10 intersections. Your system should allow creation of different configurations (using a rectangular grid) with equal distances between them. You can also assume the speed limit though variable is constant throughout the domain at any given time. The main purpose of the system is to determine near optimal traffic signal timing (the right amount of time for red, green and yellow) at each intersection at a given time (experimentally) to allow smooth flow of traffic without accidents. (You may make the necessary assumptions). This project requires good use of OO principles as well as simple animation and graphics.





Requirements

- Allow users to build rectangular grids
- Allow user to specify the frequency of cars entering the roads such as 1 car every 3 time units
- Allow users to specify the unit of time for green light in both directions in time units (assume amber is a fixed number of units)
- A car should be allowed to move if the traffic light ahead is red or it is blocked (by a car ahead)

Simplifying Assumptions

- Assume the speed of each car is fixed – such as moving a car by half a car each time unit
- Assume cars are not allowed to turn
- Each junction has 4 interconnected traffic lights (green and amber along one direction corresponds to red along the perpendicular direction.
- Cars need not be modelled once reaching the end of the road

Bonus Marks (one of the following)

- Allow users to specify turning cars – you may assume there are two lanes along each road
- Automatically determine near optimal traffic signal timing for a given configuration.

Possible Classes

Car, CarQueue, TrafficLight, Road, Grid class, Place class (a road may be divided into many places – the amount of space needed to hold a car).

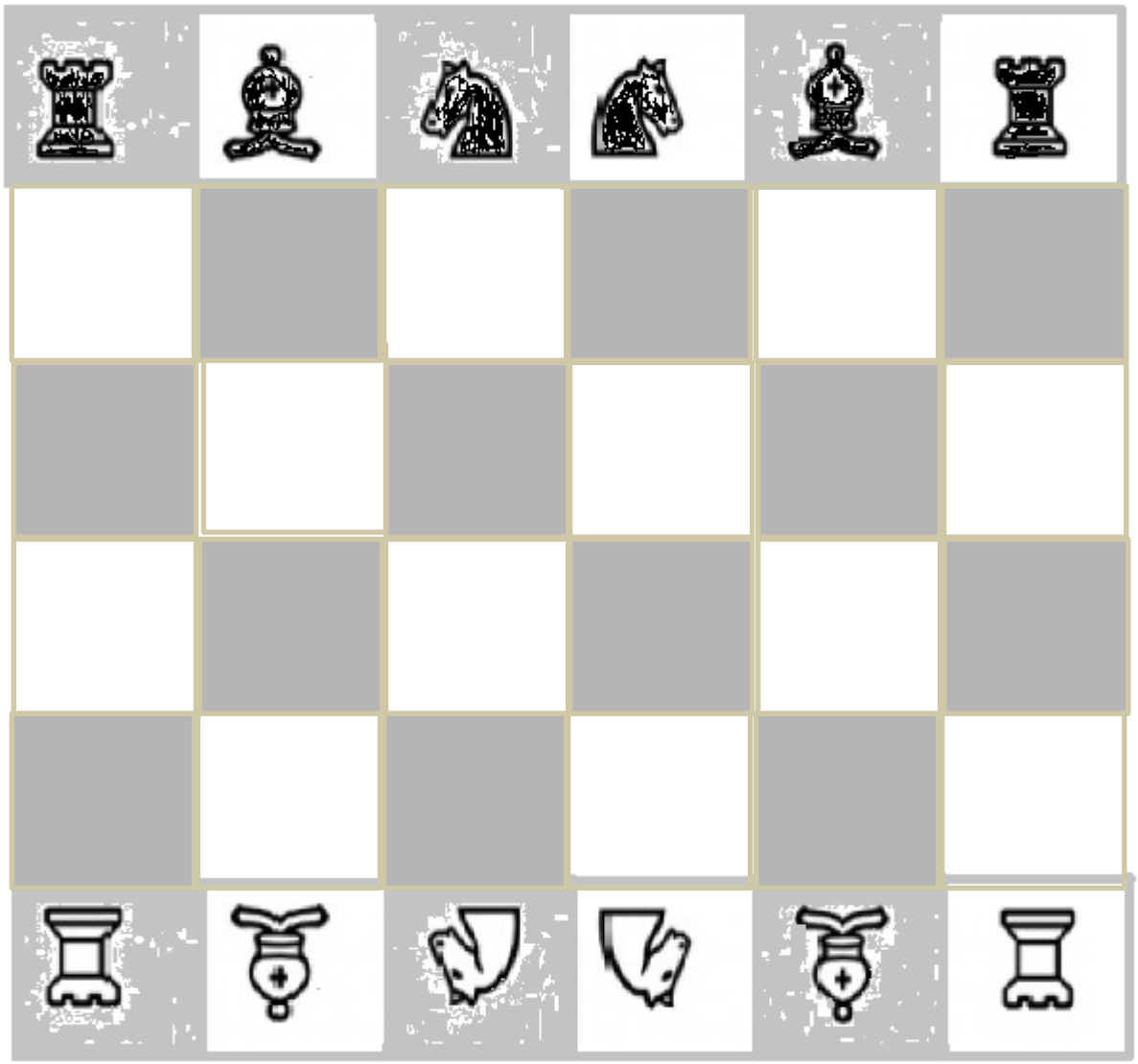
Note: You need not show a GUI version for milestone 1

Suggested Initial Test Cases for Milestone 1

1. Verify the light changes from red to amber to green and from green to amber to red
2. Verify the traffic light for cars coming in the opposite direction in any junction is the same
3. Verify the traffic lights along crisscrossing roads in the same junction are either green and red or amber and red.
4. Verify that no car moves through a red light.
5. Verify that the first car waiting at the junction starts moving into the junction only when the light turns green and junction is free of traffic.
6. Verify that no car moves over another car
7. Verify the number of cars entering at the beginning of the road is the same as the number of cars exiting at the road.

Project 6: Chess Like Game

You are required to develop a new strategic game similar to Chess, being made up of pieces such as rooks, bishops and knights. The bishop can move up to two pieces but only diagonally. The rook can move up to two pieces but only vertically or horizontally. Rook and bishop pieces cannot move over other pieces. A knight moves two squares along horizontal or vertical direction and another square perpendicularly making an "L" shape. A knight can also move over other pieces. Whenever a player removes an opponent's piece 5 points are collected. The game stops when the one play takes out all the pieces of the opponents or after n moves. The player with the most number of points is the winner.



Other Requirements

1. All players must be registered
2. Both players must be logged in before they can commence.
3. The two players must be different.
4. The maximum number of moves must be determined averaging the desired number specified by both players before the game commences.

Enhancements for Milestone 2

- Allow different kinds of pieces of the same player to merge thus combining their moving capabilities, or split when such capabilities are no longer needed. Both merging and splitting can be done only when it is the player's turn and is considered as one move.
- Show all valid moves a piece can take graphically.

Suggested Initial Test Cases for Milestone 1

1. Attempt to register two members with the same login ID fails.
2. Attempt to login with invalid login ID and password fails
3. Attempt to login with valid login ID and password succeeds
4. Attempt to login with the same valid ID and login fails
5. Attempt to move a bishop piece diagonally up to two cells, succeeds when there are no pieces.
6. Attempt to move a bishop piece diagonally bypassing other pieces fails.
7. Attempt to move a bishop piece diagonally more than two cells fails when there are no pieces.
1. Attempt to move a bishop piece in any direction other than diagonally fails
8. Attempt to move a rook piece in horizontal or vertical directions up to two cells succeeds when there are no pieces
9. Attempt to move a rook piece in horizontal or vertical directions bypassing other pieces fails
10. Attempt to move a rook piece in any direction other than horizontally or vertically fails
11. Attempt to move a knight piece along L shape (2 + 1) succeeds
12. Attempt to move a knight piece to any other cell fails
13. Attempt to move a knight piece over another piece succeeds
14. Player points are increased by 5 whenever an opponent piece is removed

Note: You need not show a GUI implementation for milestone 1 (you may display the pieces in a console – use the characters b k r and B K R to represent the white and black Bishop, Knight and Rook respectively).

Bonus Marks (choose 1)

- Provide a warning to the player if a move is likely to cause a loss of a piece in the next move by the opponent. The player can choose to ignore the warning.
- Extend it into a multiplayer (network) game
- Allow the last two moves to be undone (each player is given one chance to revert to the previous state if they make a careless move).

Project 7: Vacation House Exchange Club

Design and implement a House Exchange club which allows members to use houses of other members when they go on vacation on a weekly basis (Friday to Friday). There is an initial entry fee of $1000 when registering as a member, which earns the new member 1000 credit points. Initial registrations must capture the following information:

Member Details

- Name (personal)
- Address (personal)
- Phone number (personal)

Member House Details

- City and suburb
- Distance to city centre
- Availability of public transport
- Number of rooms
- Air-conditioning/heating facilities
- Swimming pool
- Optional video/images of the house
- Points needed for a week (Friday to Friday)
- Discounts for long stays (over multiple weeks)

Each member earns points when the house is let to another member, which can subsequently be used when the member wants to travel to other cities.

Ratings

The system will track two separate ratings which can vary from -10 to +10. The utility rating will reflect the utility level derived based on the average of responses of other members who have occupied the house in the past. In addition, users may also specify a fair week-value for the property in terms of credit points and write a review. The occupier-rating is derived by averaging the ratings of all owners who had let their houses to be used (how well that member has taken care of the house).

Letting the House

Any member can let his or her house by specifying the period and the minimum occupier (member) rating for potential occupants. The minimum occupier rating can thus be used to prevent the house being rented to poor occupants or those for which no history is available.

Viewing Houses to Let

Any member can list the details of all eligible houses in a specified week and city. The listing should consist of only those houses for which the member has adequate credit points and occupier rating. The information should also include the utility rating and the averaged-fair-value reported by previous occupants. Members should also be given the option to view the reviews on any of the listed properties.

Occupancy Request

Any member can request to occupy any property for which they are eligible. When a request is made, the owner will be able access the rating of the interested occupier and may choose to accept or reject the request. No other member can make a request while a current request is being considered. If no response is sent out within 24 hours it is considered to be rejected. If the property owner responds positively, the contact details will be revealed to the requestor after deducting the points, allowing occupier to make the necessary arrangements.

Non-Members

Non-members can view all house details (but not their availability) to encourage non-members to join.

Suggested Initial Test Cases

1. Verify Member and House details added are captured correctly
2. Verify new members are given 1000 credit points
3. Verify non-members can only view property details but not availability.
4. Verify a member can only view properties for which he or she meets the occupier-rating specified
5. Verify that the occupier rating is the average of ratings specified by all previous owners
6. Verify that utility rating is the average of ratings specified by all occupiers
7. Verify credit points for the occupier member is decreased correctly after a transaction
8. Verify credit points for the owner member is increased correctly after a transaction
9. Verify the status of a property is put on hold when a request for occupation is made
10. Verify the status of a property put on hold reverts to available if owner does not respond within 24 hours
11. Verify the status of the property is changed after a successful transaction
12. Verify that the fair value is based on the average value estimated by previous occupiers

Enhancements for Milestone 2

1. Allow members to specify their vacation requests and allow interested owners to respond
2. Allow all data to be persisted (Database or ORM tools)
3. Incorporate a map for the house

Bonus (one of the following)

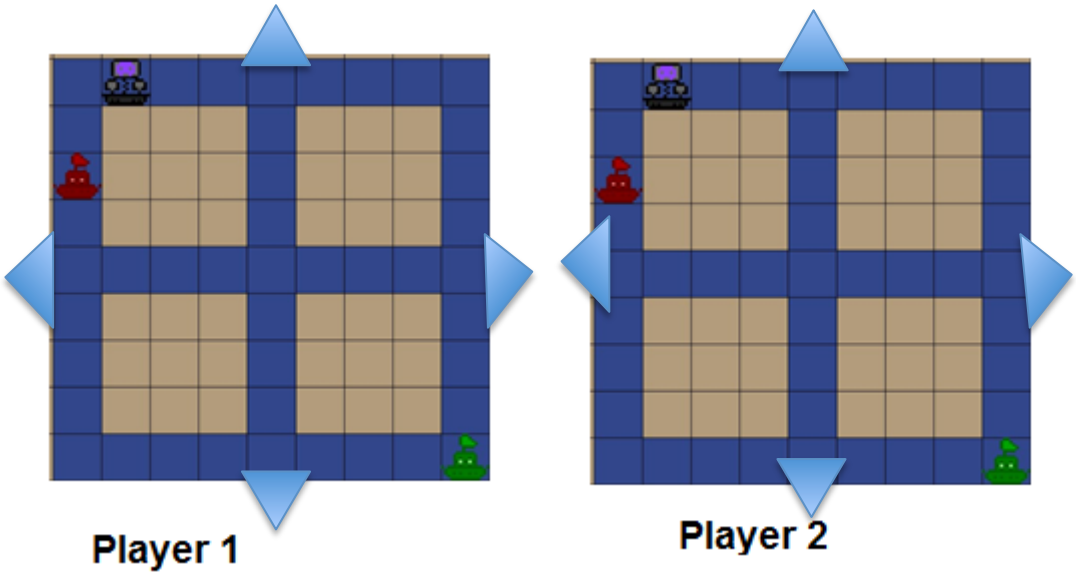
- Encrypt all data stored in the database and incorporate options for payment
- Use of design patterns (at least 5)

Specific requirements

Note that while this option is web based, it must use java/groovy on grails for your implementation with additional frameworks (ie. a pure php implementation is not acceptable) and your solution must use Object Orientation.

Project 8: Multiplayer Interactive Monster Game

You are required to write a simple multiplayer game where a monster (controlled by the server) chases and eats up the player pieces moving in the predefined grid shown below. The players and monster can also move along the arrows shown below to end up along the opposite edge. The aim of the game is to escape from the monster and to get all the opponents eaten up by blocking their escape routes. Each player should be allowed to move using appropriate arrow keys. The first player starting the game should be allowed to specify the number of players (between 2 and 4). The game should start when the required number of players joins in. Each player should be allowed to specify one of the four remaining corners as the starting cell (top-left, top-right, bottom-left, bottom-right) in the order they join the game. The monster should initially be placed in the cell at the centre and be made to move towards the nearest player (least number of cells in between). When two or more players are at equal distance, the monster may choose to move towards any one of them randomly. A player can block any other player by being stationed along the escape route (with the aim of getting the other player eaten). A monster must wait for a period of time for the food to digest before starting the chase again. The game should stop when only one player is left, who is considered to be the winner. You may use either sockets and threads (using your own protocol) or Java RMI (easier) to implement the game.



Suggested Initial Test Cases

1. Verify that the four arrow keys moves the payers in the corresponding directions (up, right, left and down).
2. Verify that players cannot move outside the grid
3. Verify that monster does not move outside the grid
4. Verify that player positions in different clients are identical (requires game logic in server to be broadcasted to all clients)
5. Test that players cannot move over each other (server must synchronize move events)
6. Verify that monster always moves towards the nearest player
7. Verify that the game does not start until the specified number of players join
8. Verify that the game stops when only one player is remaining

Enhancement

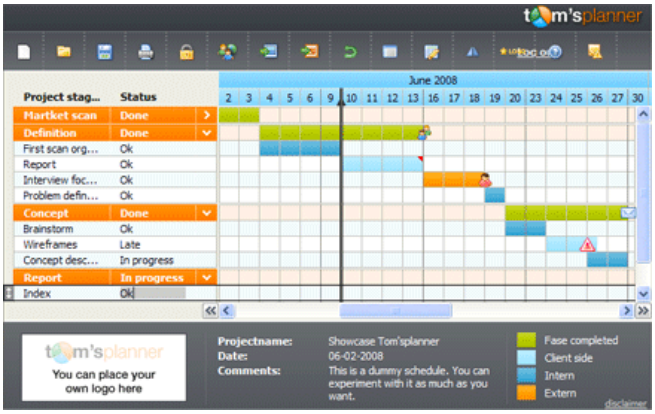
1. Allow Player Registration and Login
2. Track player scores and allow top 10 players to be displayed based on the percentage of games they win.

Bonus

1. Allow the board grid to be created at run-time.

Project 9: Task and Staff Scheduler

Assume a project manager has approached you to develop a customised project management software over the summer vacation. He does not insist on a GUI based version (but prefers one) as he is more interested in solving his immediate problems before the new year starts. He has a number of projects lined up and he wants to reduce the overall costs. This project is ideal for any student interested in project management – it also has interesting interactions between different project, activity and employee classes and will allow you to develop some heuristic algorithms.



Requirements

Jack (the Project Manager): Currently I am allocating staffs to activities in an ad hoc manner resulting in project delays and cost overruns. I need you to design and develop a Task and Staff Scheduler that assists me to complete the projects I manage within time and budget. The project involves two parts: scheduling activities and allocating staffs to activities. For each activity I can estimate the duration in weeks, the type of skills needed and the number of staff needed. For each such activity I can also specify any dependencies on previous activities (if any). The critical path and the earliest project completion time are computed based on dependencies on previous activities and the duration of activities. My main problem however, is allocating staffs to activities as staffs may be working on a number of projects in parallel; staff in our companies can be working either 20%, 40%, 60% 80% or 100% on a project reflecting the number of days per week they are engaged in the projects. I select the staffs for an activity based on whether they have all the skills needed. Very often full-time staff are underutilized in specific weeks while in some other weeks we had to hire many contract staffs. I would like to minimize the number of contract staffs as they expect premium rates to be paid on an hourly basis. This year, I want a new system that will help me plan all the activities for all the projects at the beginning of the year and maximise the usage of our permanent staff. I want the system to compute the expected project completion date based on the start date for the project (first dummy activity in the project), the duration and dependencies (predecessors) for each activity. For each activity, I also estimate the average number of staffs required (can be fractional) and their required competency levels in needed skills. Secondly, I would like to have the option to move the activities not lying along the critical-path and to experiment assigning different staff with the intention of minimizing the dependencies on external staff.

I would like your system to come up with some initial staff assignment using some simple heuristics, such as assigning the least experienced full-time staffs first to activities (as more experienced staff can fit into many different and complex activities). After assigning all the

full-time staffs, assign inexpensive, less experienced contractors first. I would still like to have an option to change these staffs around manually to do some trial-and-error.

Jim a Full-time developer: I would like a system where I can specify at the beginning of the year the weeks that are blocked (I am not available due to annual leave or other duties). I would also like to specify weeks where I m partially blocked off as I have a fair amount of non-project related duties. In such weeks I would like to specify my availability as 20%, 40%, 60%, 80% and 100%. Though I am multi-skilled my level of competency and experience vary significantly. I would like to specify my competency level explicitly for different skills in a scale of 1 to 10, so that the manager can select me for activities where I can be really productive. For example, I am highly competent in Java (10) but I am weak in Python (4) and domain modelling (4).

Timothy an External Contractor: In the past we had to charge high rates as there were a number of weeks in a row where we got no work at all followed by intensive periods. Most contractors including myself prefer to work on an activity lasting for at least two weeks or more. We tend to be more productive and happy when working for a longer period, and are even willing to charge less. I would therefore like the system to allow me specify different rates for contracts lasting 1 week, 2 weeks and 4 weeks or more.

Possible Domain Classes

Project, Activity, Staff, Contract Staff, Full-time staff, Competency, BlockedWeeks, Skill.

Bonus Requirements

Compute near optimal schedules by minimizing the involvement of contract staffs and maximizing the utilization of full-time staff.

Suggested Initial Test Cases

1. Verify there are no cyclic dependencies in the activities specified, such as activity B cannot start until A, C cannot start until B and A cannot start until C.
2. Verify that no activity in any of the projects starts before its dependent (predecessor) activities are complete.
3. Verify the earliest project completion time is computed correctly.
4. Test the commitment of a staff to an activity is either 20%, 40%, 60%, 80% or 100%.
5. Test that no staff assignment for an activity results in a staff being assigned more than 100% commitment in all the projects combined at any one week.
6. Verify that the staff skill levels are in the range 1 to 10.
7. Verify that in every activity assigned to a staff, the staff competency level is greater than or equal to the required competency level in every skill required for that activity.
8. Verify partial availability of full time staff are taken into account during project assignment.
9. Verify that any attempt to assign a staff to activity in a week that he or she is unavailable is rejected.

Project 10: Collaborative Snakes and Ladders Game

The purpose of this assignment is to create a Collaborative Snakes and Ladders game. The game introduces additional rules and objects to make it a more strategic collaborative game. You are free to add new rules to make the game more playable.

Standard Snakes and Ladders Game

Snakes and Ladders Game was originally created to depict the struggle faced by humans through good and evil forces as they go through life. The ladders represent moving up to a higher moral plane with the help of benevolent beings, while going down the snakes represent being dragged down to a more immoral level through evil powers. Snakes and Ladders game was designed for 2 or more players and is played on a board with 100 squares numbered 1 to 100. Play begins on square 1 and finishes on square 100. Players take turns to roll a dice and move along the number rolled. If a player rolls a *6*, the player may, after moving, immediately take another turn; otherwise play passes to the next player in turn. If a player lands on a square that has the bottom of a ladder upon it, the position is automatically advanced to the top of the ladder. Similarly if a player lands on a square, which has the head of a snake upon it, then the player must automatically follow down to the tail of the snake. The winner is the player who is first to land on square numbered 100. You must roll the exact number needed to land on 100.

Collaborative Snakes & Ladders (to be analysed, designed and implemented by your team)

In this version the game is played between collaborative players trying to attain the location 100 (utopia) for any one of the players within their limited timespan (30 moves allowed for players) aided by ladders and being dragged down by snakes. The game stops when one of the players reaches 100 (a win for the players) or when the timespan (30 moves for players) has run out without reaching 100 (a loss for the players).

Game Initializations

Initially the game initiator should allow customisation of the board, by specifying the positions of the 5 ladders, 5 snakes and the number of players (2-4). The ladder top positions should be greater than the ladder tail positions. The snakehead positions (greater than snake tail positions) cannot overlap with ladder top and base positions. Assume all player pieces are initially set to 1.

Game Play

This game is a turn-based game where the collaborative players, collaborative snakes and collaborative ladders get to move in turns with specific intentions. Depending on the turn, either a **ladder base** position, **snakehead** position or the **player piece position** is altered. Any of these entities can opt not to make a move when the dice value is thrown, however the number of turns for players will be incremented by 1 whether or not the turn is skipped. Whenever the dice value of 6 is thrown all entities should be allowed to throw again and add up the dice values.

Turn of Collaborative Snake objects (evil forces)

When it is the turn of the snakes any one of the snakehead can be moved in the positive or negative direction by the dice value thrown. For example, if a dice value of 4 is thrown when it is the turn of the snakes, the snake with head position 87 can move to 91 or 83 provided there no ladder tops or bases are placed on those locations. If the head of any snake ends on top of a player piece, the piece moves to the tail of the snake unless that piece is in possession of snake-escape points, in which cases snake-escape points will be reduced by 1for that piece and the snake is prevented from moving (snakes lose their turn). If a player piece is eaten up, the piece ends up at the snake tail position and a new snake should be randomly created and added (the same rules apply as in the initialization). If the position the snakehead has landed has multiple pieces all those pieces will be moved down to the snake tail and new snakes created for each piece eaten up unless any of those the piece have snake-escape points.

Turn of Collaborative Ladder objects (benevolent forces)

When it is the turn of the ladders any one of the ladder-base can be moved in the positive or negative direction by the dice value thrown with the aim of benefitting the players move up the board. For example, if a dice value of 4 is thrown when it is the turn of the ladders, the ladder with tail position 37 can move to 41 or 33. If a snakehead happens to be in one of these positions, the ladder base can be moved to that position while the snake with head at that location is eliminated (the number of remaining snakes reduced by 1). If a ladder base is moved to the position of a player piece(s) those piece(s) will be moved to the top of the ladder with one snake-escape point added to each of the piece moving up the ladder.

Turn of Collaborative Player Objects

When it is the turn of the players, any of the current players can be moved and if lands on a ladder base then the piece is moved up to the top and a snake escape point is added to the piece. The players need not move any piece; however the number of moves should be incremented by one.

Who is this Project suitable for?

This project is suitable for those who like to have an interest in graphics and multimedia but lacking the knowledge of underlying graphics (sample code provided for drawing the entities snakes, ladders, pieces and the board). This project therefore allows students to focus on the OO design instead of the underlying GUI constructs (Swing). You will be expected to pick up some basic Java Swing but the sample code should be adequate for most of the functionality.

Other Requirements

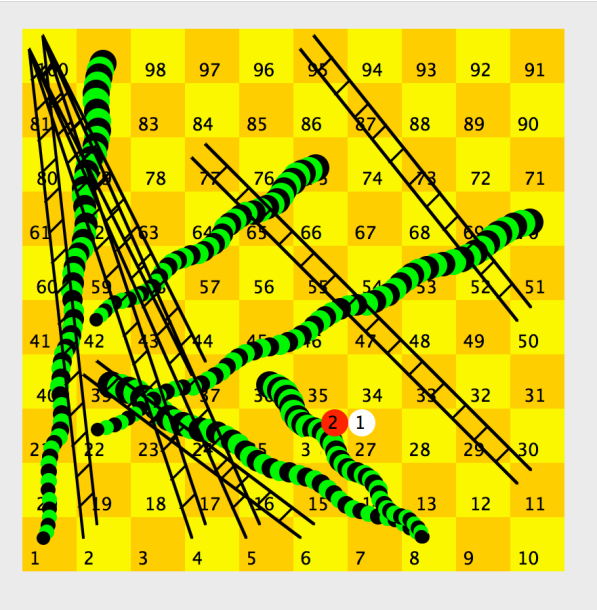
Allow users to choose a standard board or a customised board.

Allow a login feature for the players (only the leader needs to register)

Keep track of the highest scorers.

Bonus: (both of the following)

- The strategy for the ladders and snakes must be automated allowing the game to be played without inputs for moves of ladders and snakes with the objective of making the players win and lose the game respectively.
- Allow the game state to be saved and retrieved.

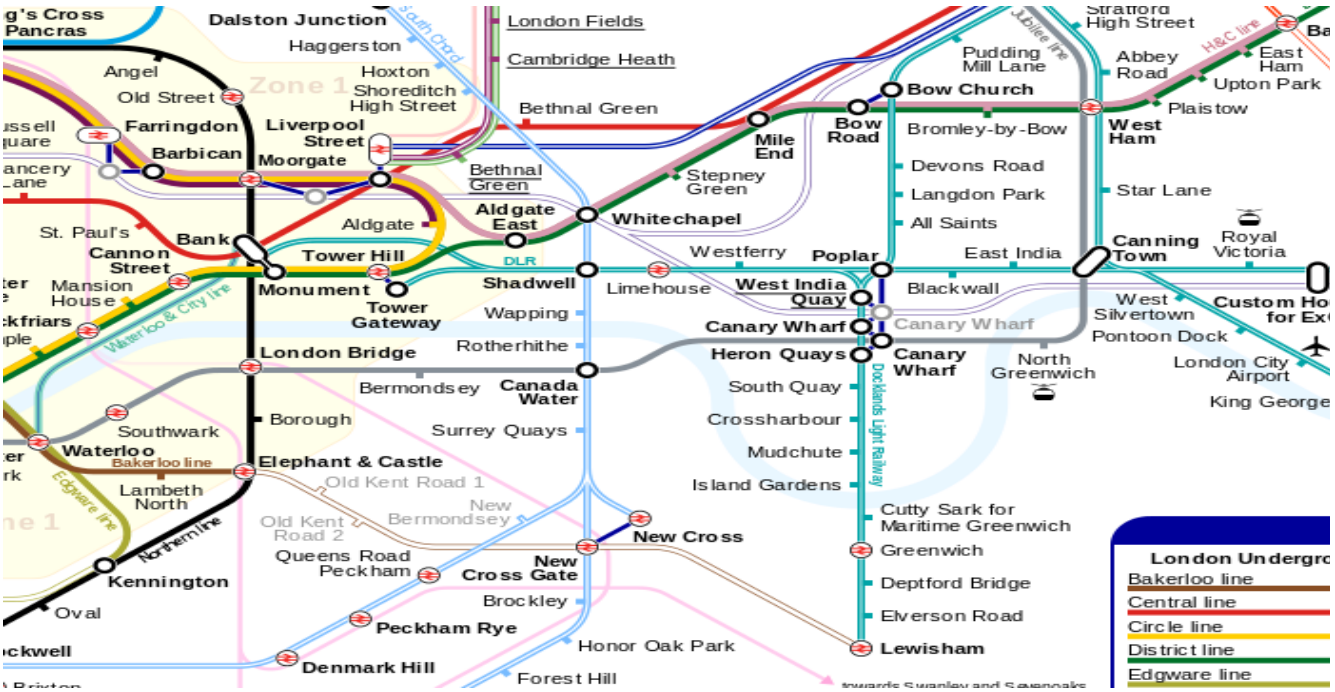


Suggested Initial Test Cases for Milestone 1

- Verify the dice can be thrown again when a 6 is thrown.
- Verify that that the SnakeEscape points of a piece is incremented each time it moves up the ladder and goes down by one when it escapes being eaten up by a snake.
- Verify an entity (player piece, ladder or snake) can only move based on the dice value thrown. Note snake and ladder entities can move in the positive or negative direction subject to specific constraints but player pieces can only move in the positive (increasing) direction.
- Verify the snakes and ladders do not overlap when the board is customised.
- Verify that the game starts when a valid (registered) user ID and password combination is supplied
- Verify that the game does not start when invalid user-ID and password combination is supplied
- Verify the highest scores are maintained correctly.
- Verify that a snake is added when a piece is eaten up.
- Verify the number of remaining snakes goes down by one when a ladder base is moved over a snakehead.
- Verify the number of moves for players (including the ones skipped) do not exceed 30.
- Verify the rules for winning and losing is correctly implemented.

Project 11. The eXcel-lent Commuter project

The client wants an online tool that helps the **eXcel London conference** visitor to use the **London tube** and **Docklands Light Railway(DLR)** network to get to the venue. The client wants the tool to use **published timetable** data for the relevant networks. The client may want the end user to enter **special needs** and **preferences** and have their **route recommendations** tailored accordingly.



Project 12. The Hyde Park/Kensington Gardens Alert project

Hyde Park and Kensington Gardens are two of the four major parks in London, linking Kensington to Buckingham Palace. Serpentine River runs through it. Every day, many people walk, commute, jog, exercise, play, visit, demonstrate or speak at Speakers Corner. There are also several daily patrols, horse training by royal guards, military exercises. Your client, representing security for HRH Queen Elizabeth II, wants a solution that can effectively provide snapshot of royal and government personnel available who can respond to a number of major incidents in this popular park.

Project 13. The RA Summer Exhibition project

For their well known Summer Exhibition this year, the Royal Academy provides visitors with an art listing booklet as part of their entry fee. A growing number of visitors complained that while the booklet is useful for showing **art description** and basic details, it has no pictures and it's not easy to find which room a piece of art may be found. The actual art display is also inconsistent whether it correctly displays a 'red dot' sticker for **sales**. The client wants a solution to address all these issues, **improve the experience** of visitors and stop printing booklets for next year. The client encourages innovative ideas.

Project 14. The Conference Tracker project

Conference staff, exhibitors and general visitors to the this year's TECHXLR8 conference at eXcel London are issued barcoded conference ID tags to recorded at entry after security checks. The client would like to make more use of this ID to collect data on different conference activities such as visitor engagement at exhibition booths and stalls, attendance at a lecture or workshop, or even purchases at food and beverage locations.

Project 15. The Kinderdojo project

The client wants an application to let parents to keep an eye on their children's progress at Kinder. Parents, carers and Kinder management will use this application to communicate easily and effectively; for instance they can share updates, learning progress and pictures.