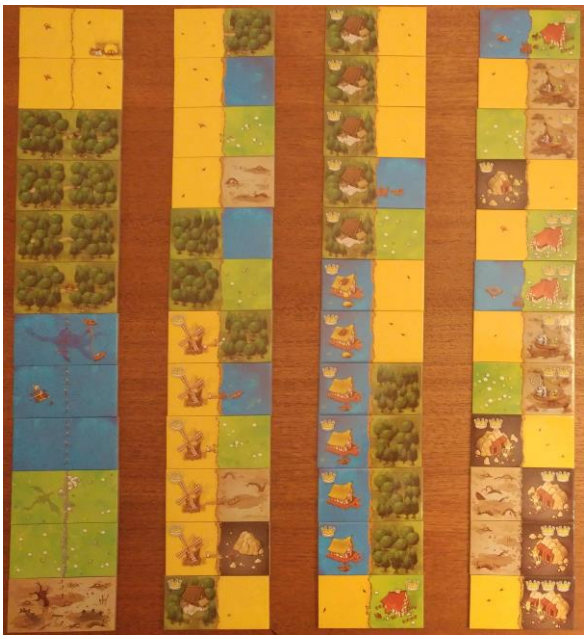
Project Overview

You will create the ***Kingdomino*** application for the well-known board game *Kingdomino*. The detailed rules of the game are uploaded to MyCourses, thus they are not repeated as part of this document.



Players are identified by their color (green, blue, yellow, pink). There are six types of terrains, namely, *wheat field* (W: yellow), *lake* (L, blue), *forest* (F: dark green), *grass* (G: light green), *mountain* (M: black), *swamp* (S: light brown).

Dominos are uniquely identified by their number (between 1 and 48) and their actual content (i.e. type of terrain on each tile and the number of crowns). The figure on the right presents all the dominos in an increasing order (i.e. top-left is 1, bottom-right is 48). For test files, these dominos will be represented textually. For example, **`19:F+W(1)`** denotes domino #19 containing a wheat field with one crown and a forest without a crown (thus the default value of crowns is 0 if not indicated otherwise).

```
1:W+W      13:W+F     25:W+F(1)     37:L+G(1)
2:W+W      14:W+L     26:W+F(1)     38:W+S(1)
3:F+F      15:W+G     27:W+F(1)     39:G+S(1)
4:F+F      16:W+S     28:L+F(1)     40:W+M(1)
5:F+F      17:F+L     29:G+F(1)     41:W+G(2)
6:L+L      18:F+G     30:W+L(1)     42:L+G(2)
7:L+L      19:F+W(1)  31:W+L(1)     43:W+S(2)
8:L+L      20:L+W(1)  32:F+L(1)     44:G+S(2)
9:L+L      21:G+W(1)  33:F+L(1)     45:W+M(2)
10:G+G     22:S+W(1)  34:F+L(1)     46:S+M(2)
11:G+G     23:M+W(1)  35:F+L(1)     47:S+M(2)
12:S+S     24:W+F(1)  36:W+G(1)     48:W+M(3)
```

1 Deliverables

You will deliver the ***Kingdomino*** application in several iterations throughout the term. For each iteration, one or more deliverables are to be submitted as described below. More detailed information will be provided for each iteration. The project is done in a ***team of six students***. If the number of students in the class is not divisible by six, then some teams may consist of five or seven students with the instructor's permission. This permission will only be granted once the final number of students is known after the add/drop deadline on **Tuesday, January 21, 2020**. Please take into account that groups of five or seven students are exceptions and not the norm.

In general, the whole team is responsible for each deliverable. For Deliverables 2-4, individual team members are responsible for some specific features.

Deliverable 1: Domain Model, UI Mockups (8%) (due Sunday, February 9, 2020 23:30)

- Use Umple to define the domain model showing all concepts and relationships of the Kingdomino game including all features to be developed to play the game.
- Capture the 10 most important (interesting) constraints of the domain which are not covered by your domain model in a controlled natural language.
- For each feature of the game, you will need to create a mockup of the user interface and document it in the wiki of your Github repository.
- Generate code from the domain model and commit the code to your group's repository in the GitHub organization of the course (https://github.com/McGill-ECSE223-Winter2020/). Further instructions will be provided on how to join your Github group.

Deliverable 2: Specification of Controller interfaces, Implementation of Controller methods, Mapping of Gherkin scenarios (10%) (due Sunday, March 1, 2020 23:30)

- As an input for this deliverable, you will receive user stories and scenarios written in the Gherkin language, which capture the main interactions during the game.
- Assign the development of each feature to one of your team members, i.e., each team member is individually responsible for four specific features.
- For each feature individually specify the Controller interface as needed to realize the feature.
- Implement the assigned features individually and as a team in Java as described in Iteration 2 and commit the code to your group's repository in the course's GitHub organization.
- For each feature, individually map the corresponding Gherkin scenarios to acceptance tests to be executed by Cucumber in the context of your Controller interface. All related acceptance tests should successfully execute upon completing this deliverable.

Deliverable 3: State Machine Specification and Implementation, Updated Controller Interface, Mapping Gherkin Scenarios (8%) (due Sunday, March 22, 2020 23:30)

- In this iteration, we provide Gherkin specification of the key phases of the gameplay as an input.
- As a team, use Umple to define the state machine to control the flow of the Kingdomino game, i.e. to initialize the game, to proceed to next turn, to proceed to next player, to proceed to next phase (choosing next domino, placing domino to kingdom), and by evaluating score. Moreover, you are also allowed to use state machines to capture the behavior of other domain classes.
- Generate code from the state machine(s) and commit the code to your group's repository in the course's GitHub organization.
- Implement the Controller part of the assigned methods of the state machine (related to events, actions, guards) as a team in Java and commit the code to your group's repository in the course's GitHub organization
- As a team, map the corresponding Gherkin scenarios to acceptance tests to be executed by Cucumber in the context of your statemachine and the underlying domain model.

Deliverable 4: Final Application (10%) (due Sunday, April 5, 2020 23:30)

- Develop a fully functional application for your Kingdomino application by developing a graphical user interface (GUI) fully integrated with the previously implemented controller methods and statemachines.
- You are responsible for the correct GUI behavior of your assigned features both individually and as a team.
- Commit the code to your group's repository in the course's GitHub organization.
- You should ensure that all related acceptance tests are still executed successfully.
- As a team, correct any mistakes in the implementation from earlier iterations

Deliverable 5: Live Demo and presentation (4%)

- Give a live demo of your application to the instructor and to your fellow students.
- Highlight the unique aspects of your application in an accompanying presentation.
- The upload of presentation slides of the demo is due on **Sunday, April 5, 2020, 23:30**.
- Demo slots will be available on **April 7** (TUE), **April 9** (THU) and **April 10** (FRI) during the lecture and tutorial slots.

2 User Stories for Key Features of the Game

As an initial input, user stories are provided for the key features of the game. However, we strongly recommend that you check the shared rules of the Kingdomino game.

Stage 1: Initialization and general game features

F#1: Set game options: As a player, I want to configure the designated options of the Kingdomino game including the number of players (2, 3 or 4) and the bonus scoring options.

F#2: Provide user profile: As a player, I wish to use my unique user name in when a game starts. I also want the Kingdomino app to maintain my game statistics (e.g. number of games played, won, etc.).

F#3: Start a new game: As a Kingdomino player, I want to start a new game of Kingdomino against some opponents with my castle placed on my territory with the current settings of the game. The initial order of player should be randomly determined.

F#4: Browse domino pile: As a player, I wish to browse the set of all dominos in increasing order of numbers prior to playing the game so that I can adjust my strategy.

F#5: Shuffle domino pile: As a player, I want to play have a randomly shuffled pile of dominos so that every game becomes unique.

F#6: Load game: As a player, I want to load a previously played game so that I can continue it from the last position.

F#7: Save game: As a player, I want to save the current game if the game has not yet been finished so that I can continue it later.

Stage 2: Choose next domino

F#8: Create next draft of dominos: As a player, I want the Kingdomino app to automatically provide the next four dominos once the previous round is finished.

F#9: Order and reveal next draft of dominos: As a player, I want the Kingdomino app to automatically order and reveal the next draft of dominos in increasing order with respect to their numbers so that I know which are the more valuable dominos.

F#10: Choose next domino: As a player, I wish to be able to choose a designated domino from the next draft assuming that this domino has not yet been chosen by any other players.

Stage 3: Place domino to kingdom

F#11: Move current domino: As a player, I wish to evaluate a provisional placement of my current domino by moving the domino around into my kingdom (up, down, left, right).

F#12: Rotate current domino: As a player, I wish to evaluate a provisional placement of my current domino in my kingdom by rotating it (clockwise or counter-clockwise).
F#13: Place domino: As a player, I wish to place my selected domino to my kingdom. If I am satisfied with its placement, and its current position respects the adjacency rules, I wish to finalize the placement. (Actual checks of adjacency conditions are implemented as separate features)
F#14: Verify castle adjacency: As a player, I want the Kingdomino app to automatically check if my current domino is placed next to my castle.
F#15: Verify neighbor adjacency: As a player, I want the Kingdomino app to automatically check if my current domino is placed to an adjacent territory.
F#16: Verify no overlapping: As a player, I want the Kingdomino app to automatically check that my current domino is not overlapping with existing dominos.
F#17: Verify kingdom grid size: As a player, I want the Kingdomino app to automatically check if the grid of my kingdom has not yet exceeded a square of 5x5 tiles (including my castle).
F#18: Discard domino: As a player, I wish to discard a domino if it cannot be placed to my kingdom in a valid way.

Stage 4: Evaluate score
F#19: Identify kingdom properties: As a player, I want the Kingdomino app to automatically determine each properties of my kingdom so that my score can be calculated.
F#20: Calculate property attributes: I want the Kingdomino app to automatically calculate the size of a property and the total number of crowns in that property.
F#21: Calculate bonus scores: As a player, I want the Kingdomino app to automatically calculate the bonus scores (for Harmony and middle Kingdom) if those bonus scores were selected as a game option.
F#22: Calculate player score: As a player, I want the Kingdomino app to automatically calculate the score for each of my property based upon the size of that property and the number of crowns. The total score of the player should also include the bonus score.
F#23: Calculate ranking: As a player, I want the Kingdomino app to automatically calculate the ranking in order to know the winner of a finished game.
F#24: Resolve tiebreak: As a player, I want the Kingdomino app to automatically resolve a potential tiebreak (i.e. equal score between players) by evaluating the most extended (largest) property and then the total number of crowns.

3 Bonus Features

The following list of features of the Kingdomino application can be considered as a bonus feature. All bonus features are due together with **Deliverable 4**.

- **B1: Three-player mode**: This feature supports to play Kingdomino with three persons instead of four. This does not involve the use of any new technologies, so it is an easy way for bonuses. **Algorithmic difficulty (AD): 1, Technological difficulty (TD): 1**
- **B2: Two-player mode**: This feature supports to play Kingdomino with only two persons instead of four. This does not involve the use of any new technologies, but it is slightly more complex than the three-player mode. **AD: 2, TD: 1**
- **B3: Dynasty**: This feature supports to play three Kingdomino games in a row among the same set of players, and then determine the overall winner at the end of the three rounds. **AD: 2, TD: 1**
- **B4: Hints:** This feature provides the user various hints during gameplay upon request (e.g. where to place a particular domino within the kingdom, which domino to select). **AD: 3, TD: 2**
- **B5: Computer gameplay**: This feature allows to play a four-player Kingdomino game against AI players. The teams can implement any AI-based approaches for computer play. **AD: 5, TD: 2-3**.
- **B6: Network/LAN-mode:** This feature enables to play a Kingdomino game where one (or each)

player is using a remote computer, thus network communication is necessitated. This feature can be demonstrated by running the same Kingdomino application on different computers (i.e. the program developed by one team), or to play across Kingdomino applications developed by different teams, which involves extra integration effort. **AD: 2, TD: 4**

- **B7: Use Eclipse Modeling Framework (EMF) as domain model**: This feature would use an alternate code generation technology (called Eclipse Modeling Framework, EMF) instead of Umple to derive the domain model used by your application. EMF is an open source framework, which may already be deployed to your Eclipse environment, and it is available at: https://www.eclipse.org/modeling/emf/. **AD: 3, TD: 5**
- **B8: Use Yakindu Statechart Tools for statemachine implementation**: This feature uses an industrial statechart modeling and code generation technology (Yakindu Statecharts) instead of Umple to provide the executable behavior for the required features. We have received academic licenses for Yakindu Statecharts, so please contact the instructor if you wish to address this feature. **AD: 3, TD: 5.**

Note that some of these features requires significant extra implementation efforts from your team. Moreover, bonus features do not pay well compared to compulsory features, so there is no point in implementing 5 bonus features if your team loses many marks on compulsory tasks.

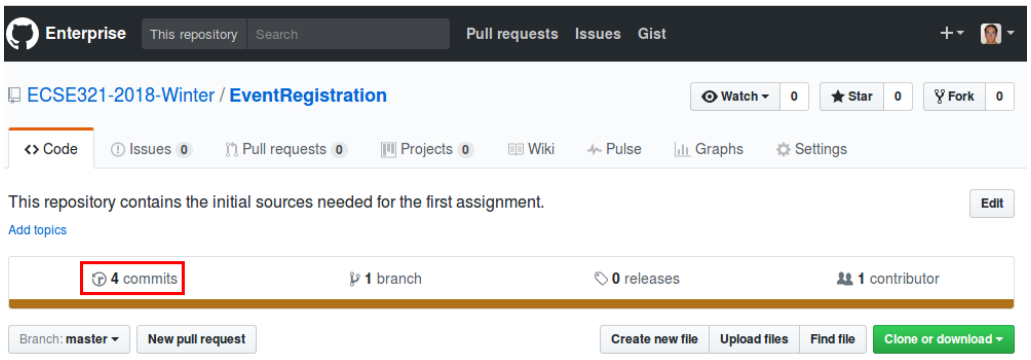
4 Technology Constraints

Your ***Kingdomino*** application must be implemented in Java with a suitable framework for the user interface. The UI does not have to be Java. However, the use of technologies other than Java Swing and Java 2D for the UI must be approved by the instructor, and they must be integrated into the automated build script (using Gradle). Your domain model and state machines must be specified with Umple and code generated from them with Umple. The use of other state machine modeling and related code generator technologies needs approval by the instructor. In all cases, **you must use the generated code in your application**. You are not allowed to make manual modifications to the generated code, but you may add native Java code to the domain model specified with Umple. Your GUI must be seamlessly integrated with the controller methods implemented in Deliverable 2 and the statemachines developed in Deliverable 3.

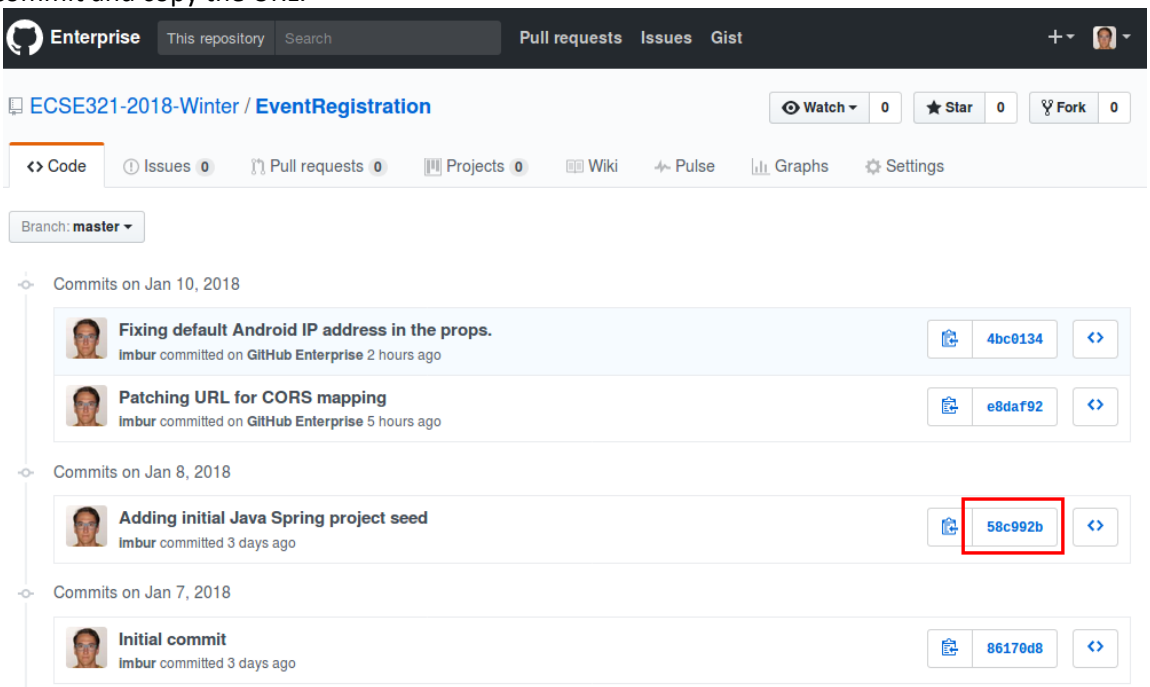
5 General Rules

Project Reports: All project reports should be provided on the wiki page of the Github repository of your group. Clearly state the course name and number, term, team number, and team members on the overview page. Each deliverable should be on a different wiki page, formatted as a Markdown document.

Submission of Source Code: Your team is required to work on the deliverables in the repository assigned to your team in the GitHub organization of this course, but **a commit link** (i.e. the URL of your final commit) **is required to be submitted in myCourses**. For that purpose, you need to select the *commits* menu for a repository:



To **get a commit link** for a specific commit, click on the button with the first few characters of the hash of the commit and copy the URL.



While it is recommended to use multiple branches in your GitHub repository, **you are required to integrate all features to the master branch when submitting a deliverable**. Contributions that exist on other branches will not be considered for grading.

Member Contributions: Each team member must contribute to each project deliverable. A team member who does not contribute to a project deliverable receives a mark of 0 (zero) for that deliverable. A team member may optionally email a confidential statement of work to the instructor ***before the due date*** of the project deliverable. A statement of work lists in point form how team members contributed to the project deliverables. In addition, the statement of work also describes whether the workload was distributed fairly evenly among the team members. A statement of work may be used to adjust the mark of a team member who is not contributing sufficiently to the project deliverable. It is not necessary to email a statement of work, if a team distributed the work for the project deliverable fairly evenly and each team member contributed sufficiently.

Note that contributions to your group's GitHub repository will be taken into account for the grade of some deliverables. You can view user activity on GitHub by opening your repository online and clicking on the Insights tab.