German University in Cairo
Media Engineering and Technology
Prof. Dr. Slim Abdennadher

Computer Programming Lab, *Spring 2019*
Rescue Simulation: **Milestone 3**

Deadline: 20.04.2019 @ 23:59

This milestone is a further *exercise* on the concepts of **object oriented programming (OOP)**. By the end of this milestone you should have added and handled all exceptions that could arise during the game play as well as implemented a Graphical User Interface for the game.

1 Graphical User Interface

You are required to implement the GUI to be able to play the game.

1.1 General Guidelines

- The action that is currently happening in the game should always be clearly indicated in the GUI.
- Make sure to handle all exceptions and validations for any input or action performed. In case any exception arises the player should be notified or the action should be prohibited and another action should be chosen by the player.
- Using a window builder or any other automated GUI creator **is not allowed**. Such submissions will receive a **ZERO**.

1.2 GUI Requirements

The requirements that should be covered in the GUI can be found in the following checklist: `https://bit.ly/2JLgdJK` . The effects of any action performed in the GUI should be reflected in the engine and vice versa.

2 Exception Classes

In this milestone you are required to deal with all the special cases and invalid actions that can arise during gameplay. In this milestone you are required to implement all exception classes with the correct messages and throw the correct exceptions in the appropriate situations. All exceptions should be thrown upon response. You should update the methods implemented in M2 by throwing the necessary exceptions preventing any illegal move from happening. Always

be sure to make the exception messages as descriptive as possible, as these messages should be displayed whenever any exception is thrown. The following section will aid you in determining where each exception should be thrown. It should be used alongside the the **Game Description Document** to revise the game rules.

3 Build the (SimulationException) Class

Name : `SimulationException`

Package : `exceptions`

Type : Class

Description : Class representing a generic exception that can occur during the simulation. These exceptions arise from any invalid action that is performed. No instances of this exception can be created.

3.0.1 Constructors

1. **SimulationException()**: Initializes an instance of a `SimulationException` by calling the constructor of the super class.
2. **SimulationException(String message)**: Initializes an instance of a `SimulationException` by calling the constructor of the super class with a customized message.

4 Build the (UnitException) Class

Name : `UnitException`

Package : `exceptions`

Type : Class

Description : A subclass of `SimulationException` representing an exception that occurs on units. No instances of this exception can be created.

4.0.1 Attributes

All class attribute is READ only.

1. `Unit unit`: The `unit` that triggered the exception.
2. `Rescuable target`: The target of the unit.

4.0.2 Constructors

1. **UnitException(Unit unit, Rescuable target)**: Initializes an instance of a `UnitException` by calling the constructor of the super class.
2. **UnitException(Unit unit, Rescuable target, String message)**: Initializes an instance of a `UnitException` by calling the constructor of the super class with a customized message.

5 Build the (DisasterException) Class

Name : `DisasterException`

Package : `exceptions`

Type : Class

Description : A subclass of `SimulationException` representing an exception that occurs on disasters. No instances of this exception can be created.

5.0.1 Attributes

All class attribute is READ only.

1. `Disaster disaster`: The disaster that triggered the exception.

5.0.2 Constructors

1. **DisasterException(Disaster disaster)**: Initializes an instance of a `DisasterException` by calling the constructor of the super class.
2. **DisasterException(Disaster disaster,String message)**: Initializes an instance of a `DisasterException` by calling the constructor of the super class with a customized message.

6 Build the (IncompatibleTargetException) Class

Name : `IncompatibleTargetException`

Package : `exceptions`

Type : Class

Description : A subclass of `UnitException` representing an exception that occurs when the target is not compatible with the current unit and thus it cannot handle it.

6.0.1 Constructors

1. **IncompatibleTargetException(Unit unit, Rescuable target)**: Initializes an instance of an `IncompatibleTargetException` by calling the constructor of the super class.
2. **IncompatibleTargetException(Unit unit, Rescuable target, String message)**: Initializes an instance of an `IncompatibleTargetException` by calling the constructor of the super class with a customized message.

7 Build the (CannotTreatException) Class

Name : `CannotTreatException`

Package : `exceptions`

Type : Class

Description : A subclass of `UnitException` representing an exception that occurs the unit is trying to rescue a target that is already safe. In order to check whether a target is already safe or not the unit should check the secondary attributes of the buildings or citizens, respectively. This check should be done **upon the unit's response** to the disaster. To make this check easier, you can add the following method to the `Unit` class: `boolean canTreat(Rescuable r)`: This method checks whether the rescuable `r` is safe or not.

7.0.1 Constructors

1. **CannotTreatException(Unit unit, Rescuable target)**: Initializes an instance of an `CannotTreatException` by calling the constructor of the super class.
2. **CannotTreatException(Unit unit, Rescuable target, String message)**: Initializes an instance of an `CannotTreatException` by calling the constructor of the super class with a customized message.

8 Build the (CitizenAlreadyDeadException) Class

Name : `CitizenAlreadyDeadException`

Package : `exceptions`

Type : Class

Description : A subclass of `DisasterException` representing an exception that occurs when the disaster is striking a citizen that is already dead.

8.0.1 Constructors

1. **CitizenAlreadyDeadException(Disaster disaster)**: Initializes an instance of a `CitizenAlreadyDeadException` by calling the constructor of the super class.
2. **CitizenAlreadyDeadException(Disaster disaster,String message)**: Initializes an instance of a `CitizenAlreadyDeadException` by calling the constructor of the super class with a customized message.

9 Build the (BuildingAlreadyCollapsedException) Class

Name : `BuildingAlreadyCollapsedException`

Package : `exceptions`

Type : Class

Description : A subclass of `DisasterException` representing an exception that occurs when the disaster is striking a building that is already collapsed.

9.0.1 Constructors

1. **BuildingAlreadyCollapsedException(Disaster disaster)**: Initializes an instance of a
   `BuildingAlreadyCollapsedException` by calling the constructor of the super class.
2. **BuildingAlreadyCollapsedException(Disaster disaster,String message)**: Initializes an instance of a `BuildingAlreadyCollapsedException` by calling the constructor of the super class with a customized message.