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

Computer Programming Lab, *Spring 2019*
Rescue Simulation
Milestone 1

Deadline: 1.3.2019 @ 23:59

This milestone is an *exercise* on the concepts of **Object Oriented Programming (OOP)**. The following sections describe the requirements of the milestone.

By the **end of this milestone**, you should have:

- A packaging hierarchy for your code
- An initial implementation for all the needed data structures
- Basic data loading capabilities from a csv file

1 Build the Project Hierarchy

1.1 Add the packages

Create a new `Java` project and build the following hierarchy of packages:

1. `controller`
2. `model.disasters`
3. `model.events`
4. `model.infrastructure`
5. `model.people`
6. `model.units`
7. `view`
8. `simulation`
9. `exceptions`
10. `tests`

Afterwards, proceed by implementing the following classes. You are allowed to add more classes, attributes and methods. However, you must use the same names for the provided classes, attributes and methods.

1.2 Naming and privacy conventions

Please note that all your class attributes must be `private` and all methods should be `public` unless otherwise stated. You should implement the appropriate setters and getters conforming with the access constraints. Throughout the whole milestone, if a variable is said to be READ then we are allowed to get its value. If the variable is said to be WRITE then we are allowed to change its value. Please note that getters and setters should match the Java naming conventions. If the instance variable is of type boolean, the getter method name starts by **is** followed by the **exact** name of the instance variable. Otherwise, the method name starts by the verb (get or set) followed by the **exact** name of the instance variable; the first letter of the instance variable should be capitalized. Please note that the method names are case sensitive.

Example 1 *You want a getter for an instance variable called* `milkCount` → *Method name =* `getMilkCount()`

2 Build the (Address) Class

Name : `Address`

Package : `simulation`

Type : Class

Description : A class representing a single cell location in the grid by its x and y coordinates.

2.1 Attributes

All the class attributes are READ only.

1. `int x`: It represents the x coordinates of the cell.
2. `int y`: It represents the y coordinates of the cell.

2.2 Constructors

1. `Address(int x, int y)`: Constructor that initializes a new cell/address by its X and Y coordinates.

3 Build the (Rescuable) Interface

Name : `Rescuable`

Package : `simulation`

Type : Interface

Description : Interface containing the methods available for all `Rescuable` objects.

4 Build the (Simulatable) Interface

Name : `Simulatable`

Package : `simulation`

Type : Interface

Description : Interface containing the methods available for all `Simulatable` objects.

DISASTERS

5 Build the (Disaster) Class

Name : `Disaster`

Package : `model.disasters`

Type : Class

Description : A class representing a `Disaster` that can take place at a `ResidentialBuilding` or occur to a `Citizen`. No objects of type `Disaster` can be instantiated. All `Disaster`s are `Simulatable` objects.

5.1 Attributes

All the class attributes are READ and WRITE unless otherwise specified.

1. `int startCycle`: The `Disaster`'s cycle that it starts from. This attribute is READ ONLY.
2. `Rescuable target`: The target being currently in the `Disaster`. This attribute is READ ONLY.
3. `boolean active`: It indicates if the `Disaster` is currently active or inactive. Initially set to FALSE.

5.2 Constructors

1. `Disaster(int startCycle, Rescuable target)`: Constructor that initializes a `Disaster` object with the given `startCycle` and `target`. Initially the `Disaster` is inactive.

6 Build the (Collapse) Class

Name : `Collapse`

Package : `model.disasters`

Type : Class

Description : A subclass of `Disaster` representing a `Collapse` that can take place at a `ResidentialBuilding`.

6.1 Constructors

1. `Collapse(int cycle, ResidentialBuilding target)`: Constructor that initializes a `Collapse` object with the given `cycle` and `target`.

7 Build the (Fire) Class

Name : `Fire`

Package : `model.disasters`

Type : Class

Description : A subclass of `Disaster` representing a `Fire` that can take place at a `ResidentialBuilding`.

7.1 Constructors

1. `Fire(int cycle, ResidentialBuilding target)`: Constructor that initializes a `Fire` object with the given `cycle` and `target`.

8 Build the (GasLeak) Class

Name : `GasLeak`

Package : `model.disasters`

Type : Class

Description : A subclass of `Disaster` representing a `GasLeak` that can take place at a `ResidentialBuilding`.

8.1 Constructors

1. `GasLeak(int cycle, ResidentialBuilding target)`: Constructor that initializes a `GasLeak` object with the given `cycle` and `target`.

9 Build the (Infection) Class

Name : `Infection`

Package : `model.disasters`

Type : Class

Description : A subclass of `Disaster` representing an `Infection` that can occur to a `Citizen`.

9.1 Constructors

1. `Infection(int cycle, Citizen target)`: Constructor that initializes an `Infection` object with the given `cycle` and `target`.

10 Build the (Injury) Class

Name : `Injury`

Package : `model.disasters`

Type : Class

Description : A subclass of `Disaster` representing an `Injury` that can occur to a `Citizen`.

10.1 Constructors

1. `Injury(int cycle, Citizen target)`: Constructor that initializes an `Injury` object with the given `cycle` and `target`.

Rescuables

11 Build the (CitizenState) Enum

Name : `CitizenState`

Package : `model.people`

Type : Enum

Description : An enum representing the different states of a `Citizen`.
Possible values are: DECEASED, RESCUED, IN_TROUBLE and SAFE.

12 Build the (Citizen) Class

Name : `Citizen`

Package : `model.people`

Type : Class

Description : A class representing one `Citizen` in the map/grid. All `Citizen`s are `Simulatable` and `Rescuable` objects.

12.1 Attributes

All the class attributes are READ and WRITE unless otherwise specified.

1. `CitizenState state`: It represents the current state of the `Citizen`.
2. `Disaster disaster`: It represents the current `Disaster` on a `Citizen`. This attribute is READ ONLY.
3. `Address location`: The `Address` of the `Citizen` on the grid.
4. `String nationalID`: It represents the `nationalID` of the `Citizen`. This variable is READ ONLY.
5. `String name`: It represents the name of the `Citizen`. This attribute is READ ONLY.
6. `int age`: It represents the `age` of the `Citizen`. This attribute is READ ONLY.
7. `int hp`: It represents the health points of the `Citizen`. It starts with value 100.
8. `int bloodLoss`: Indicates the amount of blood lost by the citizen. It starts with value 0.
9. `int toxicity`: Indicates the level of toxicity of a citizen. It starts with value 0.

12.2 Constructors

1. `Citizen(Address location, String nationalID, String name, int age)`: Constructor that initializes a `Citizen` object with the `location` address, the `nationalID`, `name` and `age`. The `state` is intially set to SAFE.

13 Build the (ResidentialBuilding) Class

Name : `ResidentialBuilding`

Package : `model.infrastructure`

Type : Class

Description : A class representing one `ResidentialBuilding` in the map/grid. All `ResidentialBuildings` are `Simulatable` and `Rescuable` objects.

13.1 Attributes

All the class attributes are READ and WRITE unless otherwise specified.

1. `Address location`: The `Address` of the `ResidentialBuilding`. This attribute is READ ONLY.
2. `int structuralIntegrity`: The `structuralIntegrity` of the `ResidentialBuilding`. Every `ResidentialBuilding` starts with `structuralIntegrity` a with value 100.
3. `int fireDamage`: Indicates the level of damage done to the building caused by fire. It starts with value 0.
4. `int gasLevel`: Indicates the level of gas in a building caused by a gad leak. It starts with value 0.
5. `int foundationDamage`: Indicates the amount of damage the foundation of the building suffers from.It starts with value 0.
6. `ArrayList<Citizen> occupants`: The list of citizens currently in the building. This attribute is READ ONLY.
7. `Disaster disaster`: The disaster (if any) currently happening in the building. This variable is READ ONLY.

13.2 Constructors

1. `ResidentialBuilding(Address location)`: Constructor that initializes a `ResidentialBuilding` object with the `location` address.

UNITS

14 Build the (UnitState) Enum

Name : `UnitState`

Package : `model.units`

Type : Enum

Description : An enum representing the different states of a `Unit`.
Possible values are: IDLE, RESPONDING, TREATING.

15 Build the (Unit) Class

Name : `Unit`

Package : `model.units`

Type : Class

Description : A class representing one `Unit` of the available rescue units in the game. No objects of type `Unit` can be instantiated. All `Unit`s are `Simulatable` objects.

15.1 Attributes

All the class attributes are READ and WRITE unless otherwise specified.

1. `String unitID`: The `Unit`'s id. This attribute is READ ONLY.
2. `UnitState state`: It represents the current state of a `Building`.
3. `Address location`: The current location/cell of the `Unit`.
4. `Rescuable target`: The target being currently rescued. This attribute is READ ONLY.
5. `int distanceToTarget`: The distance from the unit to the target. This attribute is neither READ nor WRITE.
6. `int stepsPerCycle`: The number of steps the unit can move per cycle. This attribute is READ ONLY.

15.2 Constructors

1. `Unit(String id, Address location, int stepsPerCycle)`: Constructor that initializes a `Unit` object with the given `id`, `address`, and `stepsPerCycle`. The `state` is initially set to IDLE.

16 Build the (PoliceUnit) Class

Name : `PoliceUnit`

Package : `model.units`

Type : Class

Description : A subclass of `Unit` representing a `PoliceUnit`. No objects of type `PoliceUnit` can be instantiated.

16.1 Attributes

1. `ArrayList<Citizen> passengers`: The citizen passengers in police car.This attribute is neither READ nor WRITE.
2. `int maxCapacity`: The max capacity of citizens the police car can take.This attribute is READ ONLY.
3. `int distanceToBase`: The distance from the police car to the police base. This attribute is READ and WRITE.

16.2 Constructors

1. `PoliceUnit(String id, Address location, int stepsPerCycle, int maxCapacity)`: Constructor that initializes a `PoliceUnit` object with the given `id`, `address`, `stepsPerCycle` and the `maxCapacity`.

17 Build the (FireUnit) Class

Name : `FireUnit`

Package : `model.units`

Type : Class

Description : A subclass of `Unit` representing a `FireUnit`. No objects of type `FireUnit` can be instantiated.

17.1 Constructors

1. `FireUnit(String id, Address location, int stepsPerCycle)`: Constructor that initializes a `FireUnit` object with the given `id`, `address`, and `stepsPerCycle`.

18 Build the (MedicalUnit) Class

Name : `MedicalUnit`

Package : `model.units`

Type : Class

Description : A subclass of `Unit` representing a `MedicalUnit`. No objects of type `MedicalUnit` can be instantiated.

18.1 Attributes

All the class attributes are neither READ nor WRITE.

1. `int healingAmount`: The amount by which the `MedicalUnit` heals a `Citizen`. It is set to 10. This corresponds to increasing the `hp`
2. `int treatmentAmount`: The amount by which the `MedicalUnit` treats a `Citizen`. It is set to 10. This corresponds to decreasing the `bloodLoss`.

18.2 Constructors

1. `MedicalUnit(String id, Address location, int stepsPerCycle)`: Constructor that initializes a `MedicalUnit` object with the given `id`, `address`, and `stepsPerCycle`.

19 Build the (Evacuator) Class

Name : `Evacuator`

Package : `model.units`

Type : Class

Description : A subclass of `PoliceUnit` representing an `Evacuator`.

19.1 Constructors

1. `Evacuator(String id, Address location, int stepsPerCycle, int maxCapacity)`: Constructor that initializes an `Evacuator` object with the given `id`, `location`, `stepsPerCycle` and `capacity`.

20 Build the (FireTruck) Class

Name : `FireTruck`

Package : `model.units`

Type : Class

Description : A subclass of `FireUnit` representing a `FireTruck`.

20.1 Constructors

1. `FireTruck(String id, Address location, int stepsPerCycle)`: Constructor that initializes an `FireTruck` object with the given `id`, `address` and `stepsPerCycle`.

21 Build the (GasControlUnit) Class

Name : `GasControlUnit`

Package : `model.units`

Type : Class

Description : A subclass of `FireUnit` representing a `GasControlUnit`.

21.1 Constructors

1. `GasControlUnit(String id, Address location, int stepsPerCycle)`: Constructor that initializes an `GasControlUnit` object with the given `id`, `address` and `stepsPerCycle`.

22 Build the (Ambulance) Class

Name : `Ambulance`

Package : `model.units`

Type : Class

Description : A subclass of `MedicalUnit` representing a `Ambulance`.

22.1 Constructors

1. `Ambulance(String id, Address location, int stepsPerCycle)`: Constructor that initializes an `Ambulance` object with the given `id`, `address` and `stepsPerCycle`.

23 Build the (DiseaseControlUnit) Class

Name : `DiseaseControlUnit`

Package : `model.units`

Type : Class

Description : A subclass of `MedicalUnit` representing a `DiseaseControlUnit`.

23.1 Constructors

1. `DiseaseControlUnit(String id, Address location, int stepsPerCycle)`: Constructor that initializes an `DiseaseControlUnit` object with the given `id`, `address` and `stepsPerCycle`.

24 Build the (Simulator) Class

Name : `Simulator`

Package : `simulation`

Type : Class

Description : A class representing the `Simulator` through which the player controls the `Unit`s to rescue `Citizen`s and `ResidentialBuilding`.

24.1 Attributes

All the class attributes are neither READ nor WRITE unless otherwise specified.

1. `int currentCycle`: The `currentCylce` the simulator is handling.
2. `ArrayList<ResidentialBuilding> buildings`: ArrayList of all the `ResidentialBuilding`s located in the map/grid.
3. `ArrayList<Citizen> citizens`: ArrayList of all the `Citizen`s located in the map/grid.
4. `ArrayList<Unit> emergencyUnits`: ArrayList of all the available `Unit`s for resucing.
5. `ArrayList<Disaster> plannedDisasters`: ArrayList of the `Disaster`s read from the CSV file, but not yet executed.
6. `ArrayList<Disaster> executedDisasters`: ArrayList of the `Disaster`s that were planned and are now executed.
7. `Address[][]world`: A 10x10 2D array that contains all the possible addresses in the current simulation world.

24.2 Constructors

1. `Simulator()`: Constructor that initializes the `Simulator` object.The `world` array should be initialized properly such that each location in the array contains the `Address` object with the `x` and `y` values correspond to its location in the array. For example, the value of the `x` instance variable of the `Address` Object located at `world[5][6]` should be 5. The value of its `y` instance variable should be 6. The ArrayLists of `ResidentialBuilding`s, `Citizen`s, emergency`Unit`s and `plannedDisaster`s are initially loaded from csv files.

24.3 Methods

1. `private void loadUnits(String filePath)`: Reads the CSV file with `filePath` and loads the `Unit`s into the `emergencyUnits` ArrayList.
2. `private void loadBuildings(String filePath)`: Reads the CSV file with `filePath` and loads the `ResidentialBuilding`s into the `buildings` ArrayList.
3. `private void loadCitizens(String filePath)`: Reads the CSV file with `filePath` and loads the `Citizen`s into the `citizens` ArrayList.
4. `private void loadDisasters(String filePath)`: Reads the CSV file with `filePath` and loads the `Disaster`s into the `plannedDisasters` ArrayList.
   Hint: When loading the citizens, buildings and units, you should use the addresses from the `world` array as their locations instead of re-initializing them.

24.4 Description of CSV files format

1. You should add `throws Exception` to the header of any constructor or method that reads from a csv file to compensate for any exceptions that could arise.
2. **Buildings**
   (a) The buildings are found in a file titled `buildings.csv`.
   (b) Each line represents a building.
   (c) The data has no header, i.e. the first line represents the first building.
   (d) The parameters are separated by a comma (,).
   (e) The line represents the building's data as follows: **LOCATION_X, LOCATION_Y**.
3. **Citizens**
   (a) The citizens are found in a file titled `citizens.csv`.
   (b) Each line represents a citizen.
   (c) The data has no header, i.e. the first line represents the first building.
   (d) The parameters are separated by a comma (,).
   (e) The line represents the building's data as follows: **LOCATION_X, LOCATION_Y, NATIONAL_ID, NAME, AGE**.
4. **Disasters**
   (a) The disasters are found in a file titled `disasters.csv`.
   (b) Each line represents a disaster.
   (c) The data has no header, i.e. the first line represents the first building.
   (d) The parameters are separated by a comma (,).
   (e) The line represents the building's data as follows: **START_CYCLE, DISASTER_TYPE, TARGET_ID**.
      i. **TARGET_ID**: This is the nationalID for the target citizens and the location for the target buildings. In case of buildings the **TARGET_ID** consists of two values; the x and y coordinates of the location separated by a comma.
      ii. **DISASTER_TYPE**: It can be one of the following values:
         A. INJ representing Injury
         B. INF representing Infection
         C. FIR representing Fire
         D. GLK representing Gas Leak
5. **Units**
   (a) The available units are found in a file titled `units.csv`.

(b) Each line represents a unit.

(c) The data has no header, i.e. the first line represents the first unit.

(d) The parameters are separated by a comma (,).

(e) All units are initially located at Address (0,0).

(f) The line represents the unit's data as follows: **UNIT_TYPE, UNIT_ID, STEPS_PER_CYCLE**. In case of loading an evacuator, a fourth value will follow the previous three values that indicates its **CAPACITY**.

i. **UNIT_TYPE**: It can be one of the following values:

A. AMB representing an Ambulance.

B. DCU representing a DiseaseControlUnit.

C. EVC representing an Evacuator.

D. FTK representing a FireTruck.

E. GCU representing a GasControlUnit.

CONTROLLER

25 Build the (CommandCenter) Class

Name : `CommandCenter`

Package : `controller`

Type : Class

Description : A class representing the controller responsible for the communication between the model and the view classes.

25.1 Attributes

All the class attributes are neither READ nor WRITE.

1. `Simulator engine`: It represents the `Simulator` through which we can control all items on the grid.
2. `ArrayList<ResidentialBuilding> visibleBuildings`: ArrayList of all the `ResidentialBuilding`s that suffered from a disaster(s) and reported about it/them.
3. `ArrayList<Citizen> visibleCitizens`: ArrayList of all the `Citizen`s that suffered from a disaster(s) and reported about it/them.
4. `ArrayList<Unit> emergencyUnits`: ArrayList of all the available `Unit`s for rescuing the `Citizen`s or `ResedentialBuilding`s.

25.2 Constructors

1. `public CommandCenter()`: Constructor that initializes the `CommandCenter` object and intializes all of its variables.