The Underworld

2016

Contents

1 Introduction

The scope of the next assignment is to make you familiarize some more with the work flow in OOP while working with Java. You will be challenged to create a logically structured project to understand how **imports** work.

Some code was already provided to you so that you won't get stuck with useless tasks and concentrate on the things that matter. All the code you will have to write won't be more than *150* lines so don't over complicate things.

Through the challenges that you will encounter the following are worth mentioning:

- Read from the keyboard
- Read text from a file & work with try-catch
- Differentiate between static and non-static methods/variables
- Create instances of classes and work with them

2 Structure

When you load the project you will notice that there is only one package `underworld.main` that contains a class called `Game.java`. However, you will notice in the following section that you will be asked to create some more classes and that is why you will also need to create two more packages.

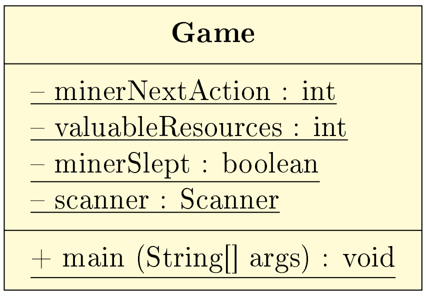
Create the following packages inside the project:
`underworld.models`
`underworld.utils`

3 Classes

In the following subsections you can find the specifications for each of the classes in your game. You will have to translate the specifications to code exactly as we did in the last laboratory. For further information regarding methods implementations please check the next section.

3.1 Package `underworld.main`

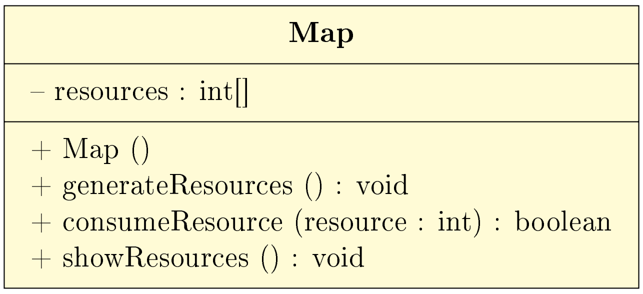
3.1.1 Game



Note: The skeleton of the project has this class partially implemented. You won't have to create a new `Game.java` file.. Just change the existing one when needed.

3.2 Package `underworld.models`

3.2.1 Map


Map
– resources : int[]
+ Map ()
+ generateResources () : void
+ consumeResource (resource : int) : boolean
+ showResources () : void

3.2.2 Miner

Miner

– name : String
– backpack : int[]
– backpackIndex : int

+ Miner (name : String)
+ getName () : String
+ setName (name : String) : void
+ dig () : int
+ sleep () : int
– isBackpackFull () : boolean
+ showBackpack () : void
– getNumberOfBackpackSlots ()

3.3 Package `underworld.utils`

3.3.1 Constants

Constants

+ WORLD_NAME : String = "your_world_name"
+ PATH_TO_RULES_FILE : String = "your_path_to_rules_file"
+ MINER_BACKPACK_SIZE : int = 5
+ MAP_RESOURCES_NUMBER : int = 10
+ NUMBER_OF_ROUNDS : String = 5

3.3.2 FileReader

FileReader

+ printFileContents (pathToFile : String) : void

3.3.3 Resources

Resources

– random : Random = new Random()

+ generateValuableResource () : int
+ generateResource () : int

4 Guidelines

4.1 Game

4.1.1 `main (String[] args)`

Take care about this method after you finished implementing the ones from other classes. The skeleton of this method is already present. In order for the game to work please **do not change** any code from here as long as you don't have any idea of what you are doing.

The only things that you will need to change in this class are the commented lines that start with the `TODO` keyword. The comments are pretty much self explanatory so you should be able to implement it without any further information.

4.2 Map

Regarding the `resources` attribute please make sure that, besides **declaring** it, you are also **initializing** it. You can do this either in the same line where you declared it or inside the body of the `Map` class.

4.2.1 `Map ()`

Inside here you should make sure you have some default values inside your `resources` array.

4.2.2 `generateResources ()`

Update the values in the `resources` array with some new values. Use the static method `generateValuableResource()` from the `Resources` class.

4.2.3 `consumeResource (int resource)`

Check whether the resource taken as parameter is found through the `resources` array. Make sure you update each of the positions in your array where the resource was found with **-1**. This method will return **`true`** if the resource was found through your `resources` array and **`false`** otherwise.

4.2.4 `showResources ()`

This method will print to console the values stored inside your `resources` array at the moment the method is called.

4.3 Miner

We will place some more logic inside the body of our `Miner` class. Please pay attention, again, to also **initialize** the `backpack` array before trying to use it. As it was the case for the `Map` class you can do this either inline where the declaration is or inside the constructor.

Note: `backpackIndex` is a variable in which we will store the current position in which we will be able to place some resource inside our miner's `backpack` array.

4.3.1 `Miner (String name)`

Update the name of the `Miner` accordingly and initialize the values from the miner's `backpack` to **-1**.

4.3.2 `sleep ()`

Inside this method a resource should be generated by calling the `generateResource` method on your `Resources` class. The value of the resource must be returned.

4.3.3 `dig ()`

Check whether the backpack is full and, if not, generate a resource the same way you did for the `sleep` method. Store the value in the `backpack` and return it.

4.3.4 `isBackpackFull ()`

Returns **`true`** if `backpack` is full and **`false`** otherwise.

4.3.5 `showBackpack ()`

Display to console the resources stored inside the `backpack` array at that moment in time.

4.3.6 `getNumberOfBackpackSlots ()`

As its name says, it will return the number of empty slots left in our miner's `backpack`. Call this method from `showBackpack` to also display the numbers of empty slots besides all the values stored inside the `backpack`.

4.4 Resources

Besides **declaring** the static attribute `random` make sure you also **initialize** it inline.

4.4.1 `generateValuableResource ()`

Generate a random number between 0 and **Constants.MAP_RESOURCES_NUMBER**; limits included.

4.4.2 `generateResource ()`

Generate a random number between 0 and **Constants.MAP_RESOURCES_NUMBER * 2**; limits included.

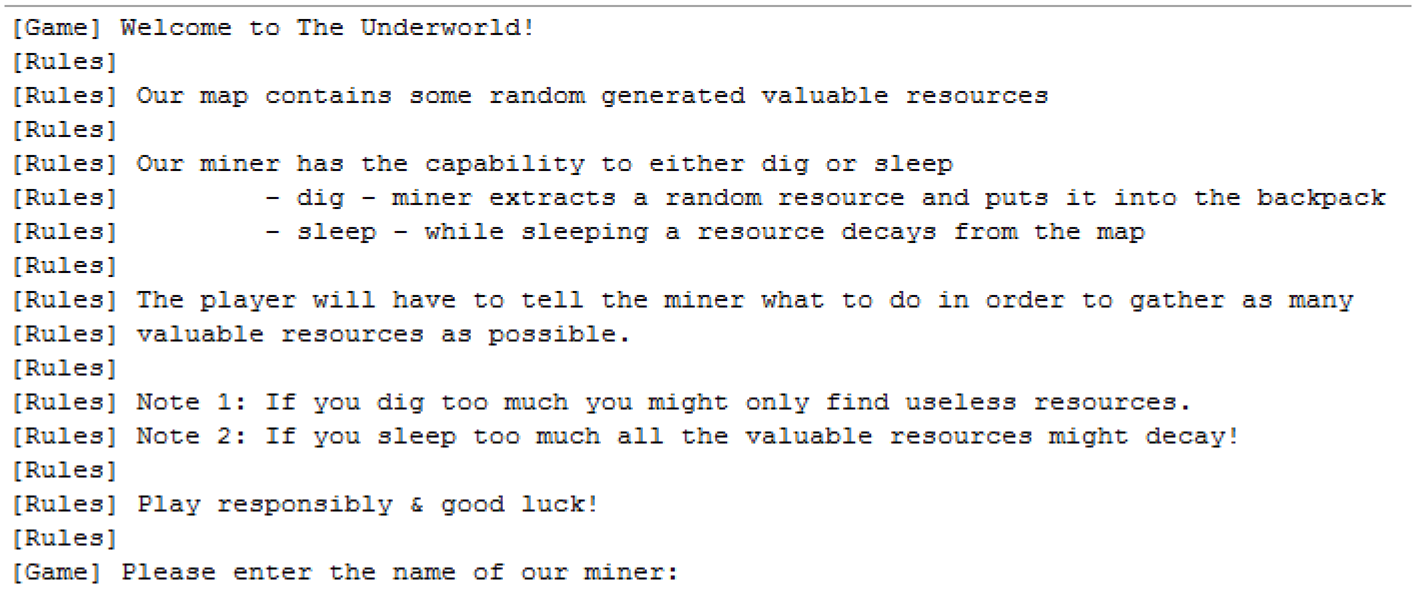
4.5 FileReader

4.5.1 `printFileContents (String pathToFile)`

This method must print to console the contents of a file located at the path specified by the `pathToFile` parameter. Search for an example of `Files.readAllLines` online. Use **`try-catch`** blocks to handle file reading.

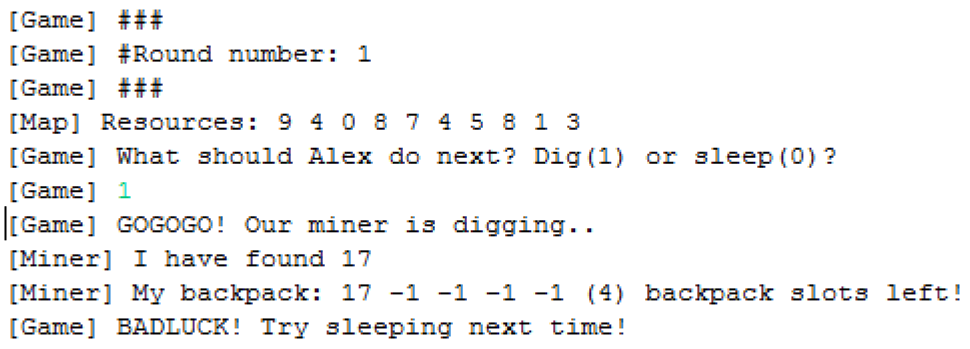
5 Final game

Below you can find some snapshots from the game I have implemented. Get creative with your messages and place some labels as I did to show where the messages are printed from. For example, if a message is printed from a method inside `Game` class you will see that the message will have `[Game]` label printed first.



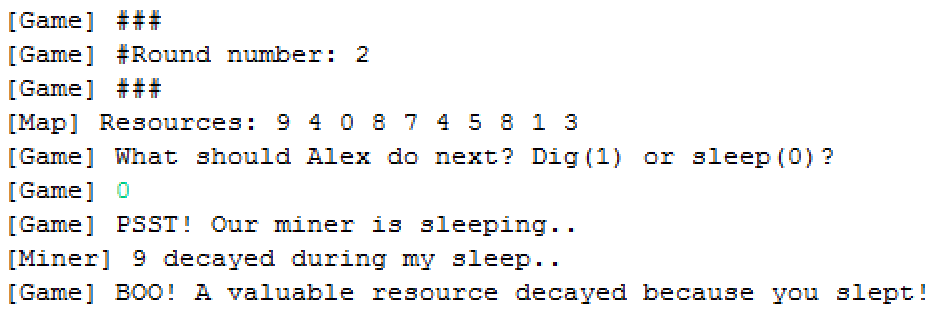
```
[Game] Welcome to The Underworld!
[Rules]
[Rules] Our map contains some random generated valuable resources
[Rules]
[Rules] Our miner has the capability to either dig or sleep
[Rules]         - dig - miner extracts a random resource and puts it into the backpack
[Rules]         - sleep - while sleeping a resource decays from the map
[Rules]
[Rules] The player will have to tell the miner what to do in order to gather as many
[Rules] valuable resources as possible.
[Rules]
[Rules] Note 1: If you dig too much you might only find useless resources.
[Rules] Note 2: If you sleep too much all the valuable resources might decay!
[Rules]
[Rules] Play responsibly & good luck!
[Rules]
[Game] Please enter the name of our miner:
```

Figure 1: Starting message taking into account the `rules.txt` file is successfully read.



```
[Game] ###
[Game] #Round number: 1
[Game] ###
[Map] Resources: 9 4 0 8 7 4 5 8 1 3
[Game] What should Alex do next? Dig(1) or sleep(0)?
[Game] 1
[Game] GOGOGO! Our miner is digging..
[Miner] I have found 17
[Miner] My backpack: 17 -1 -1 -1 -1 (4) backpack slots left!
[Game] BADLUCK! Try sleeping next time!
```

Figure 2: Messages displayed during a round the miner is digging. The numbers and `[Game]` messages might differ.



```
[Game] ###
[Game] #Round number: 2
[Game] ###
[Map] Resources: 9 4 0 8 7 4 5 8 1 3
[Game] What should Alex do next? Dig(1) or sleep(0)?
[Game] 0
[Game] PSST! Our miner is sleeping..
[Miner] 9 decayed during my sleep..
[Game] BOO! A valuable resource decayed because you slept!
```

Figure 3: Messages displayed during a round the miner is sleeping. The numbers and `[Game]` messages might differ.

6 Sources

Inside the same directory where you found this documentation you could also find the project skeleton located inside `underworld` directory and the `rules.txt` file.

Good luck & have fun!