Asteroid Mining

Team name: Pied Pipers

Supervisor:

Dr. Balla Katalin

Members:

Abdelrahman Desoki	DOHDZF	adesoki@edu.bme.hu
Neda Radonjic	NQC17Y	neda.radonjic@edu.bme.hu
Tushig Bat-Erdene	QBI3JH	bat-erdene.tushig@edu.bme.hu
Chaitanya Arora	BWTFX8	chaitanya.arora@edu.bme.hu
Janibyek Bolatkhan	BOI6FK	janibyekbolatkhan@edu.bme.hu
Kasay Ito	LX5RFB	kasay.ito@edu.bme.hu

0. Changes regarding UI catalogues

We hereby inform you that there are several changes from the last documentation which was "5.2 Plans of the skeleton's UI, dialogs."

1. In the last submission, we specified that the tester must provide the user input in order to start the game. However, in the skeleton version, this case is omitted and it automatically starts the game since this testing is redundant.
2. For the same reason described above, when the settler moves to another place, none of mouse click inputs is required. This decision can be modified and discussed later. Furthermore, "addVisitor(s1)", "removeVisitor(s1)" and "setPlace()" are not shown when the testing is conducted.
3. Mouse click inputs for selecting a resource to put/fill it into the hollowed asteroid are not implemented in this skeleton version, because GUI is not ready for it yet and this appeared to be a non-critical test case.
4. For testing of "Sunstorm occurs", "Control Asteroid Explosion" and "WaterIce sublimates", a user input X_KEY is expected before typing to the console.
5. Especially for "Control Asteroid Explosion", this test case is not validatable since there are no such asteroids that are explosive in this skeleton version. Instead, "WaterIce sublimates" case can be observed in terms of that the same function is called.

Kasay Ito (Pied Pipers)
2022/03/28

6. Skeleton program

6.1 Deployment guide

6.1.1 List of files

File name	Size	Date	Content
Asteriod.java	1.62KB	2022.03.23	The class contains the GUI and major attributes of the class Asteroid like depth, hollow, distancefromSun etc, and some functions required in rendering the file, getting resources, deepen hole etc which

			covers most of the use case related to asteroid
Carbon.java	73 Bytes	2022.03.23	Carbon class is a child class of resource
Direction.java	101 Bytes	2022.03.24	The direction is an enum containing the movements up. Down, left right, which is used to handle the movement of the spaceship
Game.java	3.31 KB	2022.03.24	The game is one of our main classes, which is initiating the game by beginning the threads when the game begins and renders all objects like asteroids, spaceships etc which need to be displayed. It contains functions like render, tick and run which control the main structure of the game.
GameObject.java	1004 Bytes	2022.03.24	Besides having the normal getter,setter. This function is used to change the movements of the objects like asteroid and spaceship and check for collisions
Handler.java	984 Bytes	2022.03.24	Handler class is for getting neighbour place, adding and removing objects,

			getting settler, and checking if the asteroid is explosive.
ID.java	230 Bytes	2022.03.24	Enumeration of Settler, Robot, Asteroid, RadioActiveAsteroid, TeleportationGate, SunStorm, Iron, Carbon, Uranium, and WaterIce.
iron.java	71 Bytes	2022.03.23	Iron class is a child class of resources.
KeyHandler.java	2.06 KB	2022.03.24	This class is responsible for creating the key handler of the game. It receives the pressed key or button to do actions such as moving, drilling and hiding.
MouseHandler.java	842 Bytes	2022.03.27	This function will be responsible for making sure the functions the user can perform via keyboard. He/she can implement those via buttons in the GUI
Place.java	764 Bytes	2022.03.26	The place class is superclass of
RadioactiveAsteriod.java	244 Bytes	2022.03.24	RadioActiveAsteroid is a superclass of Asteroid with explode method.

Resources.java	145 Bytes	2022.03.24	Class defines the type of resources.
Settler.java	1013 Bytes	2022.03.26	The settler class is the GUI of the settler and will involve the movements and basic rendering.
Robot.java	353 Bytes	2022.03.24	The robot class is the graphical user interface of the robot which involves the movements, basic rendering, and it can get damage.
Spaceship.java	239 Bytes	2022.03.24	This class is responsible for initializing spaceship with its capacity and current inventory.
Sunstorm.java	320 Bytes	2022.03.24	SunStorm class is GUI implementation sunstorm itself.
Teleportationgate.java	520 Bytes	2022.03.24	The TeleportationGate class is the graphical user interface of the gates.
Uranium.java	74 Bytes	2022.03.23	Uranium class is a child class of resource

Visitor.java	979 Bytes	2022.03.26	Visitor class extends the game object and deals with the what things can be done when a settler visit the asteroid and similar functions like drill, hide etc.
WaterIce.java	110 Bytes	2022.03.23	WaterIce class is a child class of resource. It differs with sublime() method from other resources.
Window.java	883 Bytes	2022.03.22	The class responsible for setting frame and setting window visible.

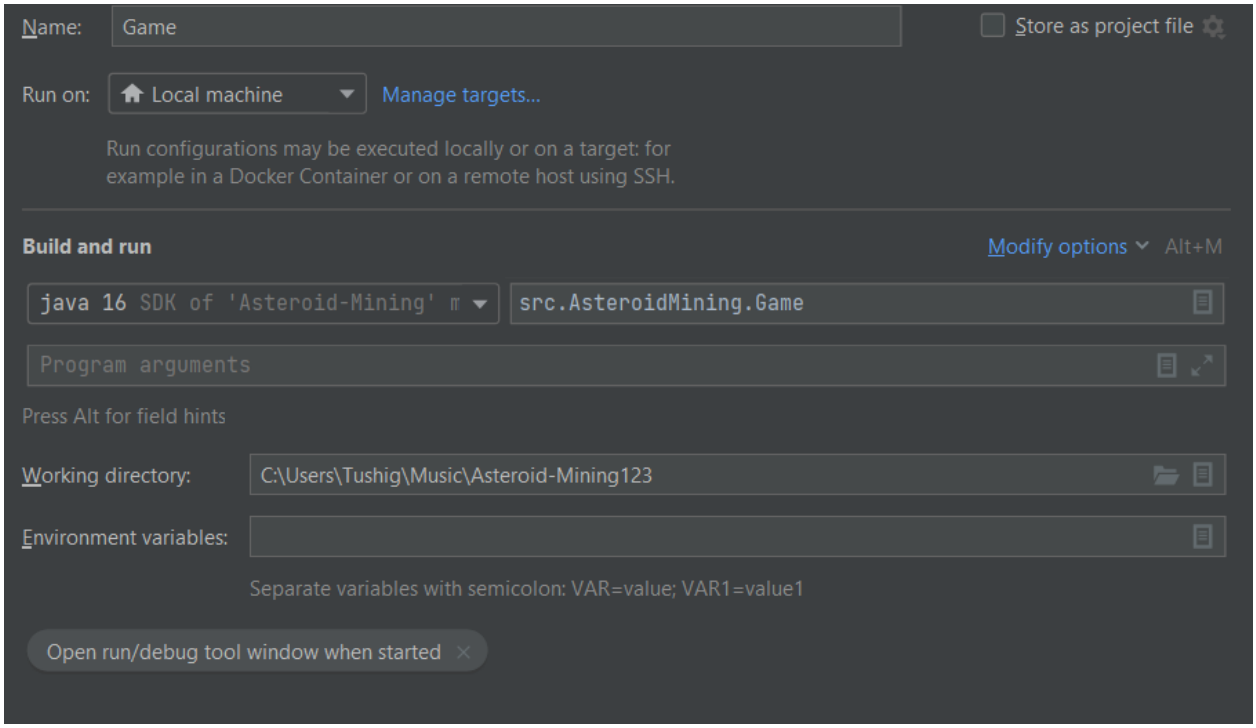
6.1.2 Compilation

In order to do the successful compilation of the code, one can clone the project or download the zip file and run it in Intellig IDE.
To execute the program we need to run the main function present in the Game.java file. Namely, the entry point is this function.
The link to github repository:
https://github.com/jackdotb/Asteroid-Mining

6.1.3 Run

There are no particular requirements except the fact that jdk should be installed in the users IDE and should preferably use intellij IDE. In order to compile the project successfully, “Game” class is needed to configurated in “Run/Debug Configuration” section of the IntelliJ IDE. Especially, some of test cases are suggested that they should be checked in debugging mode with breakpoints.

Run/Debug Configuration:



6.2 Evaluation

Name of the team member	Participation (%)
Janibyek Bolatkhan	16.66%
Kasay Ito	16.66%
Chaitanya Arora	16.66%

Tushig Bat-Erdene	16.66%
Neda Radonjic	16.66%
Abdelrahman Desoki	16.66%

6.3 Protocol

Start (date & time)	**Duration (hours)**	**Performer(s) name**	**Activity description**
22rd March	*2.5 hours*	*All team members*	We conducted an online meeting after the results were released to the mentor and discussed the next steps. These involved brushing up on your skills of version control system and making accounts on github for few of the team members. After which we tried to connect it with Eclipse, but when we encountered issues, we shifted to intellij and everyone was successfully able to connect the project to the IDE
23rd March	3 Hours	Tushig	Designed the essential graphics of the game which used to implement the graphical user interface.
23rd March	3 hours	Desoki	After the discussion, Desoki created the class structure from the help of the class diagram given in the previous submissions. As well as creating the attribute and methods.
23rd March	4 hours	Janibek	Jack started creating the rendering portion and of files from the

			experience in game development projects in the past. He was able to leverage them and create threads and rendering functions which are used in GUI
24th march	3 hours	Chaitanya Arora	Chaitanya started working on the particular functions of the settler by taking help from the sequence diagrams.
25th March	4 hours	Kasay ito	Kasay started working in the development of the backend logic and also implementing separate functions of the settler required in the execution.
26th March	2.5 hours	Neda	Neda implemented the Storyboard of how the game should look like and helped in the documentation and deciding the internal structure in the code
28th March	3 hours	Chaitanya and Tushig	After getting most of the work done, we finished all the documentation required for the documentation
28th March	6 hours	Kasay Ito	Checking all skeleton codes based on testable cases and declaring some slight modification against UI catalogs that we defined last time.