CA2 – AndroidRus

Douglas Mesquita
Student no – 2018247

Lecture – Mark Morrissey

April 2019

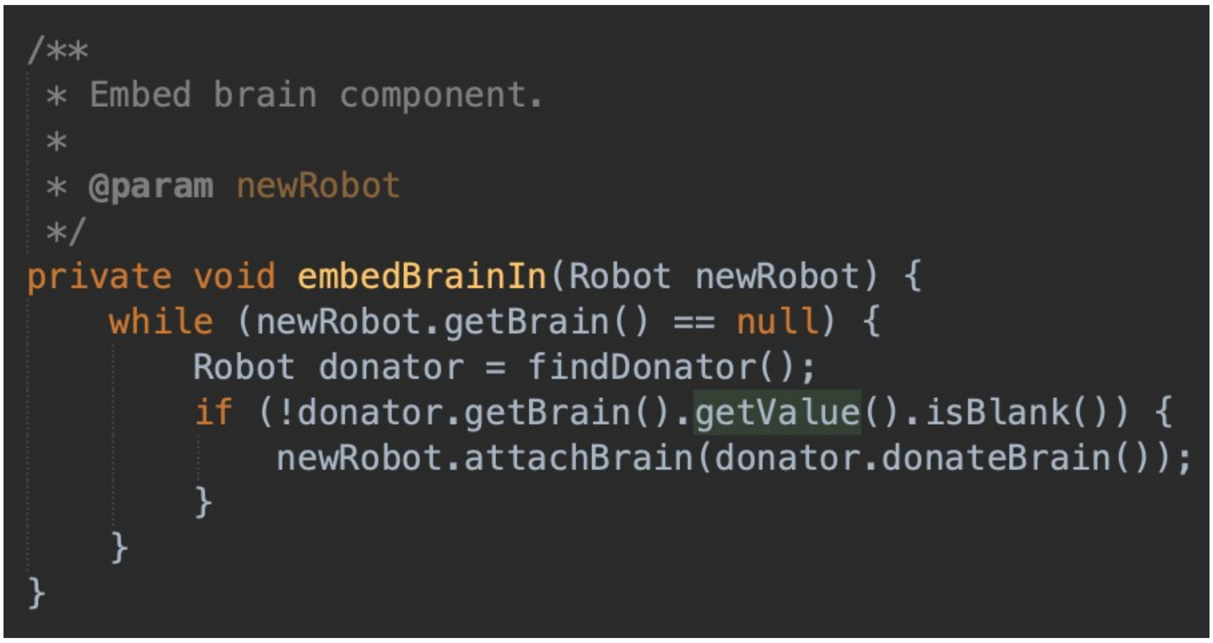
Summary

The algorithm that I've chosen to build the second generation of robots is Linear Search which is a sequential search for finding an element within a list. So the idea is getting a component from an old robot, verify if the specific component is available and then take it to the new robot.

***Linear Search complexity**

Algorithm	Best performance	Worst performance	Average performance
Linear Search	O(1)	O(n)	O(n)

The method responsible to embed a brain component into a second generation robot for instance is **embedBrainIn()**.

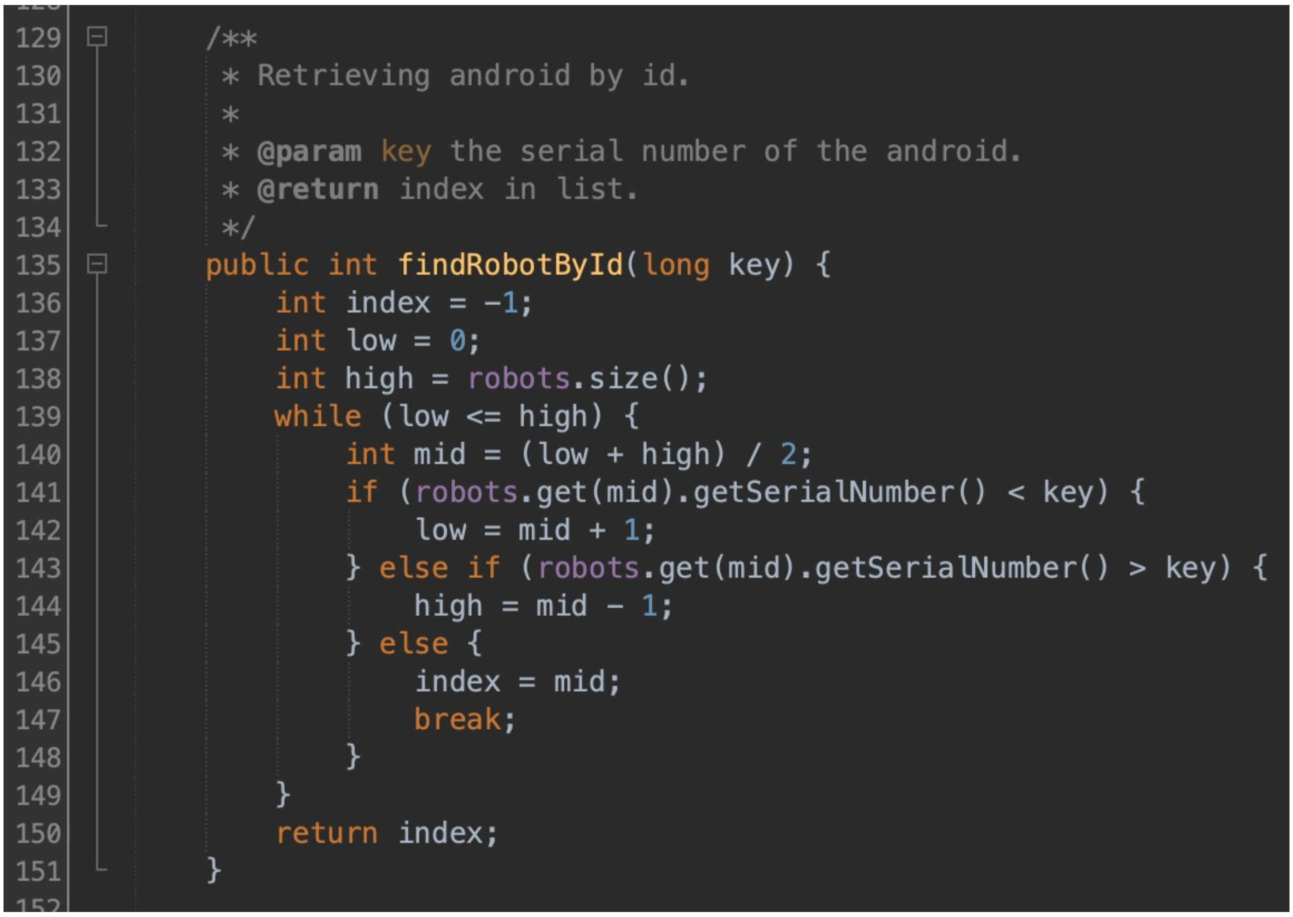


```
/**
 * Embed brain component.
 *
 * @param newRobot
 */
private void embedBrainIn(Robot newRobot) {
    while (newRobot.getBrain() == null) {
        Robot donator = findDonator();
        if (!donator.getBrain().getValue().isBlank()) {
            newRobot.attachBrain(donator.donateBrain());
        }
    }
}
```

The algorithm that I’ve chosen to do search is Binary Search. It’s a search algorithm that finds the position of a target value within a sorted array. In project, the idea is finding a robot that needs to show all information available.

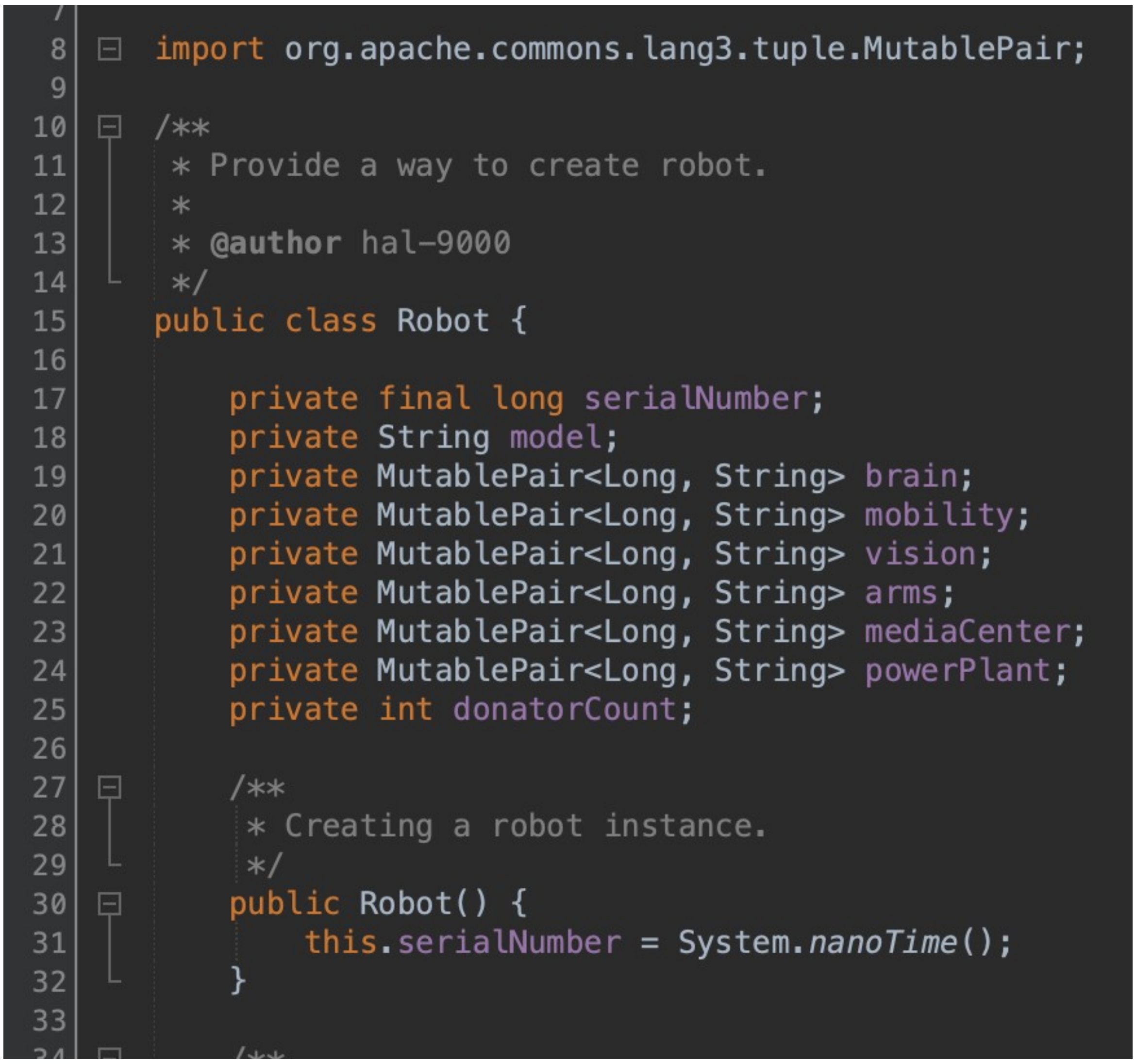
***Binary search complexity**

Algorithm	Best performance	Worst performance	Average performance
Binary Search	O(1)	O(log n)	O(log n)



```
    /**
     * Retrieving android by id.
     *
     * @param key the serial number of the android.
     * @return index in list.
     */
    public int findRobotById(long key) {
        int index = -1;
        int low = 0;
        int high = robots.size();
        while (low <= high) {
            int mid = (low + high) / 2;
            if (robots.get(mid).getSerialNumber() < key) {
                low = mid + 1;
            } else if (robots.get(mid).getSerialNumber() > key) {
                high = mid - 1;
            } else {
                index = mid;
                break;
            }
        }
        return index;
    }
```

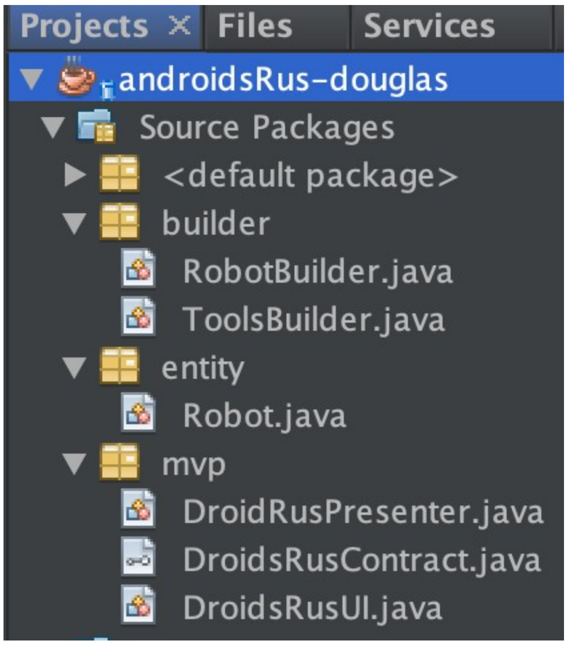
The data structures I've chosen for the project are List and MutablePair. The list is responsible to store all robots first generation and second generation. The MutablePair associate the serial number of a robot with the value (Component).



```java
import org.apache.commons.lang3.tuple.MutablePair;

/**
 * Provide a way to create robot.
 *
 * @author hal-9000
 */
public class Robot {

    private final long serialNumber;
    private String model;
    private MutablePair<Long, String> brain;
    private MutablePair<Long, String> mobility;
    private MutablePair<Long, String> vision;
    private MutablePair<Long, String> arms;
    private MutablePair<Long, String> mediaCenter;
    private MutablePair<Long, String> powerPlant;
    private int donatorCount;

    /**
     * Creating a robot instance.
     */
    public Robot() {
        this.serialNumber = System.nanoTime();
    }

    /**
```

All components are represented by mutablePair which long parameter is the serial number of the robot and String is the component type.

I’ve implement a MVP (Model view controller). It’s an architectural pattern which mostly used for building use interfaces. In MVP, all presentation logic is pushed to the presenter. Basically on project, there are three packages such as builder, presenter and mvp.



```
androidsRus-douglas
  Source Packages
    <default package>
    builder
      RobotBuilder.java
      ToolsBuilder.java
    entity
      Robot.java
    mvp
      DroidRusPresenter.java
      DroidsRusContract.java
      DroidsRusUI.java
```

*** Builder package**

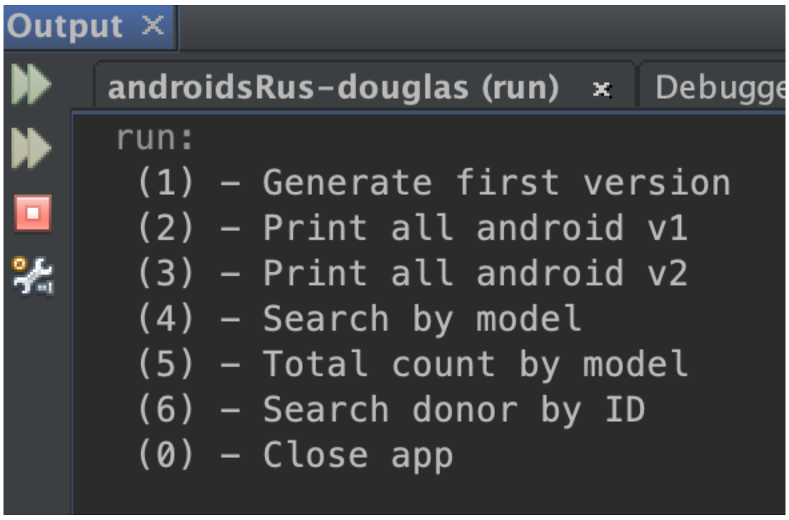
In this package, the class **RobotBuilder.java** creating robots for first generation and second. Also, **ToolsBuilder.java** creates all components available for instance: brain, mobility, arms, media center, power plant, etc.

* **Entity package**

In this package, there is only entities of the project. In that case, it’s **Robot.java.** You can check how attach a component or how to get components reference.

*** MVP package**

For this package, the class **DroidsRusContract.java** is responsible to dispatch events for presenter layer and UI layer. The **DroidRusPresenter.java** take care of business logic and **DroidRusUI.java** show all output from users.



```
run:
 (1) - Generate first version
 (2) - Print all android v1
 (3) - Print all android v2
 (4) - Search by model
 (5) - Total count by model
 (6) - Search donor by ID
 (0) - Close app
```

Option 1

Generating all robots for the first generation.

Options 2 – 3

Showing all robots for the first and second generation. In that view, it shows all information available.

Printing androidV1 - you will see some robots with no component which they already donated it to another AndroidV2. If you wish to see what robot receive the component, go to option 5.

Printing androidV2 – you will see the full information about androidV2.

Option 4

Searching a bunch of robot with the same model. Type “Android mk1” and the result show all MK1 stored.

Option 5

Showing total count by model. This feature, will show total count for all model created.

Eg:

Total Andy – 100
Total Android mk1 – 50

Option 6

Searching a donor robot by serial number. The result should show the receiver, donor and the model that donated some components.