Modular and Object Oriented Programming
Computer Science Engeneering
UPV/EHU

Project

Serengeti Natural Park

Iyán Álvarez

May 9, 2020

Contents

1. Introduction

In this document there is an explanation of the project that we have done on the MOOP subject on first grade of Computer Science Engineering at EHU/UPV university.
The programming language that we are using is Java. We have learn some other interesting tools such as JavaDoc or JUnit 5.
To develop this project I have used IntellijIDEA as IDE and Git for VCS.
This projects purpose is to work on concepts that we have been working on the subject.

- **Part 1**
  - Attributes
  - Constructors
  - Methods
  - Packages
  - Encapsulation
  - Singleton
  - ArrayLists
  - Exceptions
  - Files
  - UML
  - Junit 5 tests
- **Part 2**
  - Hierarchy
  - Inheritance
  - Polymorphism
  - Dispatching
  - Incremental programming
  - Overrides
  - Interfaces

2. Part 1

2.1 Summary

This project has been done with the objective of practising the concepts learned at Modular and Object Oriented Programming subject at first grade of Computer Science Engineering at EHU/UPV.

We are going to develop and simulate the Serengeti Natural Park, using GPS locations, Drones, Tracking Devices and Specimens. These will be controlled by Serengeti Monitor.

2.2 Designing

I have made a project diagram using the tool StarUML, organizing how we were going to implement all the project and what are the attributes, constructors, methods... that have to be implemented in each class.
I have found some aesthetic problems at the time of showing the parent package name (pack::class) or showing the Exception that throw a method, although I have added it on the specific part as was required. Despite these issues I have organized the diagram with the more dependent classes at the top (the ones that need other classes to function), and the more independent classes at the bottom (the ones that only need primitive types and default Java classes).
After finishing with the diagram, using an extension on StarUML I generated all the project "code skeleton". With this last step I finished the part of the basic design of the project.

2.3 Coding

Later, I have implemented and documented all the attributes, constructors and methods shown in the StarUML diagram for GPS, Drone, Specimen and SerengetiMonitor classes. The class TrackingDevice was given by the MOOP teachers.
Exceptions were created in another package; this is the first contact that we are having with Java hierarchy. These exceptions extend the Exception class. This means, Exception is the super-class of the classes that we are creating, these are the sub-classes. In my opinion, it is a good practice to implement a main program in each of the classes to test the classes little by little, and if an error is found, it can be easily fixed.
After finishing with the implementation and little class simulations I had changed some parts of the diagram to work better or for being clearer. *

2.4 New attributes and methods

In my opinion, I think that some attributes, methods were important to implement or necessary to made testing or software maintenance easier. These are the methods that I have personally added to the first part of the project:

- **GPS class**
  - equals()
- **Drone class**
  - private final int IMAGE_WIDTH = 1000
  - private final int IMAGE_HEIGHT = 1000
- **Specimen class**
  - public GPS initialLocation() ...
  - public void clearDayRoute() ...
  - equals();
- **SerengetiMonitor class**
  - private final int COLLECT_IMAGE_ATTEMPTS = 5

2.5 Name changes

I have changed some methods names for better comprehension.

- **Specimen class**
  - lastPosition() → lastLocation()
  - public void clearDayRoute() ...
  - equals();
- **SerengetiMonitor class**
  - closestSpecimen() → closestSpecimenList()
  - captureWithAttemps() → captureWithAttempts()
  - collectImage() → collectImagesClosest()
  - updatePositions() → updatePositionsSpecimens()

2.6 Testing

After finishing the implementation of the basic Classes, we have developed some tests for the methods that were asked to be tested. For this task we are using the tool JUnit 5, the purpose of the tests is to verify the functioning of some methods. In some cases, the expected case is manually created and we compare it with the real one, this means, the one returned by the method.
I have found some problems at the time of thinking how to propose the the tests of some methods because they were random. In this case
I have implemented 5 classes to make the tests.

- **GPSTest:** 2 cases for distanceTo()
- **SpecimenTest:** 3 cases for kmsTraveled()
- **SerengetiMonitorTestDefault:** 2 cases for closestSpecimenList()
- **SerengetiMonitorTestDrones:** 2 cases for captureWithAttempts() and other cases for collectImage(), with Drones at the park.
- **SerengetiMonitorTestNoDrones:** 2 cases for captureWithAttempts() and other 2 cases for collectImage(), with no Drones at the park.

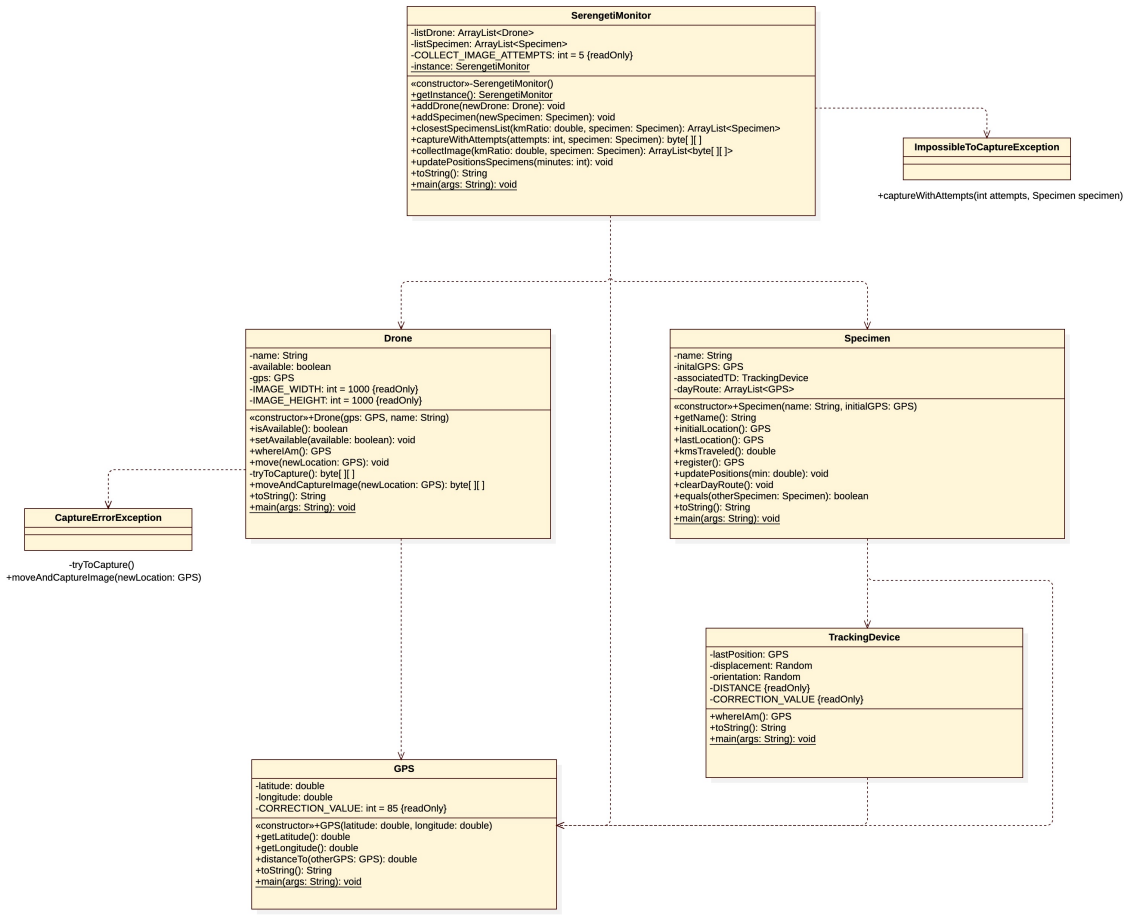
2.7 Simulating

To finish with this part of the project we have created a SerengetiSimulator class. The purpose of this class is to simulate a situation where some Drones and Specimens are on the Serengeti National Park and we can monitor what are they doing and send them actions, such as simulating the movement of all the specimens at the park, a drone moving and taking photos of a concrete specimen or knowing what are the specimens nearby a specimen in an specific ratio. Each simulation is different, some methods (the ones that simulate movement are random). Here is one of the simulations done:

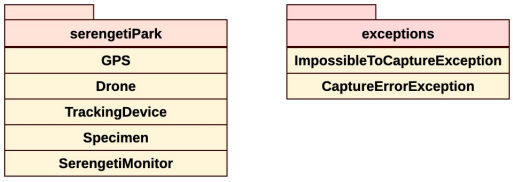
```
Trying to create SerengetiMonitor instance.
Created successfully.

Trying to add 10 Specimens to the park on the given coordinates.
Showing SerengetiMonitor status after adding 10 Specimens.
SerengetiMonitor:
 - Drones: ()
 - Specimens: (Specimen1 -2.949865 35.062473 Specimen2 -3.049865 34.962472999999996 Specimen3 -2.749865 35.262473 Specimen4 -3.249865 34.762473 Specimen5 -2.549865 35.462472999999996 Specimen6 -3.449865 34.562473 Specimen7 -2.349865 35.662473 Specimen8 -3.649865 34.362472999999994 Specimen9 -2.149865 35.862472999999994 Specimen10 -3.849865 34.162473 )

Simulating 6 hours of movement registers on all the Specimens in the park.
Showing SerengetiMonitor status after 6 hours of movement in the park.
SerengetiMonitor:
 - Drones: ()
 - Specimens: (Specimen1 -2.3473567242871005 34.979283966919226 Specimen2 -2.8490289080957005 35.26077110658032 Specimen3 -2.818781051485518 35.6641451838086 Specimen4 -3.4162430661615404 34.145691742691845 Specimen5 -2.598596007338011 35.44228795585249 Specimen6 -3.5187810514855187 34.32717888235294 Specimen7 -2.63389012498507 35.99522913232308 Specimen8 -3.7960580220140336 34.127178882352936 Specimen9 -2.6691842426321295 35.69609493383845 Specimen10 -3.8498649999999994 34.39776711764706 )

Showing the data of the Specimens in the ratio of 300km to the first Specimen added.
Specimen2 -2.8490289080957005 35.26077110658032 48.89605992632275km.
Specimen3 -2.818781051485518 35.6641451838086 70.67154681919236km.
Specimen4 -3.4162430661615404 34.145691742691845 115.21793136833166km.
Specimen5 -2.598596007338011 35.44228795585249 44.77603397816152km.
Specimen6 -3.5187810514855187 34.32717888235294 113.95948432979121km.
Specimen7 -2.63389012498507 35.99522913232308 89.72417246621036km.
Specimen8 -3.7960580220140336 34.127178882352936 142.8611698272884km.
Specimen9 -2.6691842426321295 35.69609493383845 66.7880928956429km.
Specimen10 -3.8498649999999994 34.39776711764706 136.9448124940891km.

Adding 3 drones to the park, the first one is not available.
Showing SerengetiMonitor status after adding the Drones.
SerengetiMonitor:
 - Drones: (Drone1, false, -2.949865 35.062473 Drone2, true, -2.949865 35.062473 Drone3, true, -2.949865 35.062473 )
 - Specimens: (Specimen1 -2.3473567242871005 34.979283966919226 Specimen2 -2.8490289080957005 35.26077110658032 Specimen3 -2.818781051485518 35.6641451838086 Specimen4 -3.4162430661615404 34.145691742691845 Specimen5 -2.598596007338011 35.44228795585249 Specimen6 -3.5187810514855187 34.32717888235294 Specimen7 -2.63389012498507 35.99522913232308 Specimen8 -3.7960580220140336 34.127178882352936 Specimen9 -2.6691842426321295 35.69609493383845 Specimen10 -3.8498649999999994 34.39776711764706 )

Trying to take photos of the Specimens nearby the first added Specimen in the ratio of 90km.
Drone2 moving to -2.8490289080957005 35.26077110658032 ... arrived.
Trying to take capture ... Trying to take capture ... image was taken successfully.
Drone2 moving to -2.8490289080957005 35.26077110658032 ... arrived.

Drone2 moving to -2.818781051485518 35.6641451838086 ... arrived.
Trying to take capture ... Trying to take capture ... Trying to take capture ... image was taken successfully.
Drone2 moving to -2.818781051485518 35.6641451838086 ... arrived.

Drone2 moving to -2.598596007338011 35.44228795585249 ... arrived.
Trying to take capture ... Trying to take capture ... image was taken successfully.
Drone2 moving to -2.598596007338011 35.44228795585249 ... arrived.

Drone2 moving to -2.63389012498507 35.99522913232308 ... arrived.
Trying to take capture ... image was taken successfully.
Drone2 moving to -2.598596007338011 35.44228795585249 ... arrived.

Drone2 moving to -2.6691842426321295 35.69609493383845 ... arrived.
Trying to take capture ... Trying to take capture ... Trying to take capture ... Trying to take capture ... Trying to take capture ... Specimen9 couldn't be photographed.
Drone2 moving to -2.598596007338011 35.44228795585249 ... arrived.

1 of 5 Specimens couldn't be photographed.
5 specimens are in the ratio of 90km to the first added specimen.
4 images have been taken.

Process finished with exit code 0
```

2.8 Project design

2.8.1 Classes design



2.8.2 Package design



3. Part 2

3.1 Summary

We are going to develop and simulate some specific Specimens of the Serengeti Natural Park. The main idea is to divide Specimens in different types to reuse the code as much as possible. For example, mammals, reptiles, herbivorous, carnivorous... species share some characteristics.

3.2 Designing

I have made a project diagram using the tool StarUML, organizing how we were going to implement all the project and what are the attributes, constructors, methods on each class. This tool have helped me a lot to have the main idea of the project and what is the hierarchy of Specimen. We can document each method in the StarUML tool and this will be done at the time of generating the code skeleton.
For example, 2 of 3 herbivorous make the same sound so I have decided to implement that sound() method on Herbivorous class and override the method on the other Specimen. This way we implement only 2 methods instead of making 3 different overrides on the concrete classes.
Other example is that all the felines make are carnivorous, so they have the same feeding. Is important to think and spend time on the UML, this makes us the coding task easier.
After finishing with the diagram, using an extension on StarUML I generated all the project's code skeleton. With this last step I finished the part of the basic design of the project.

3.3 Coding

Later, I have implemented all the constructors and methods shown in the StarUML diagram for all the Specimen subclasses.
The hierarchy and inheritance concepts are very important and help us a lot to reduce the amount of repeated code, categorising the specific method in each class, and all the subclasses can use the method or override it if needed.
We have used ExtinctionDanger interface to implement in the specimens needed the ability to check the danger level of each. This can be done overriding the method in the class.
The method inheritancePath() is based on incremental programming, this makes the methods shortest and easier because each one makes the specific task at each class and calls the super method if is needed.
After finishing with the coding of Specimen subclasses, I have coded the SpecimenProof class where we can have relevant information of the specimns at the park, this class is where we are going to make the simulation later.

3.4 Testing

After finishing the implementation of the basic Classes, we have developed some tests for all the concrete classes of Specimens. For this task we are using the tool JUnit 5, the purpose of the tests is to verify the correct functioning of all the concrete Specimens created, checking the expected values an the ones that each method returns.
The values that each method have to return have been defined as private static on the specific classes and can be returned using a static getter of the constant. This is done to ease software maintenance task.
I have found some problems at the time of thinking how to propose the the tests of some methods because they were random. In this case
I have implemented 3 types of tests classes to make the tests.

- **Default:** Tests getName(), feeding(), sound() and inheritancePath().
- **Mammal:** Tests the methods specified at default and walk().
- **Endangerd Specimens:** Tests the methods specified at default, in some cases mammal, and whatDangerLevel().

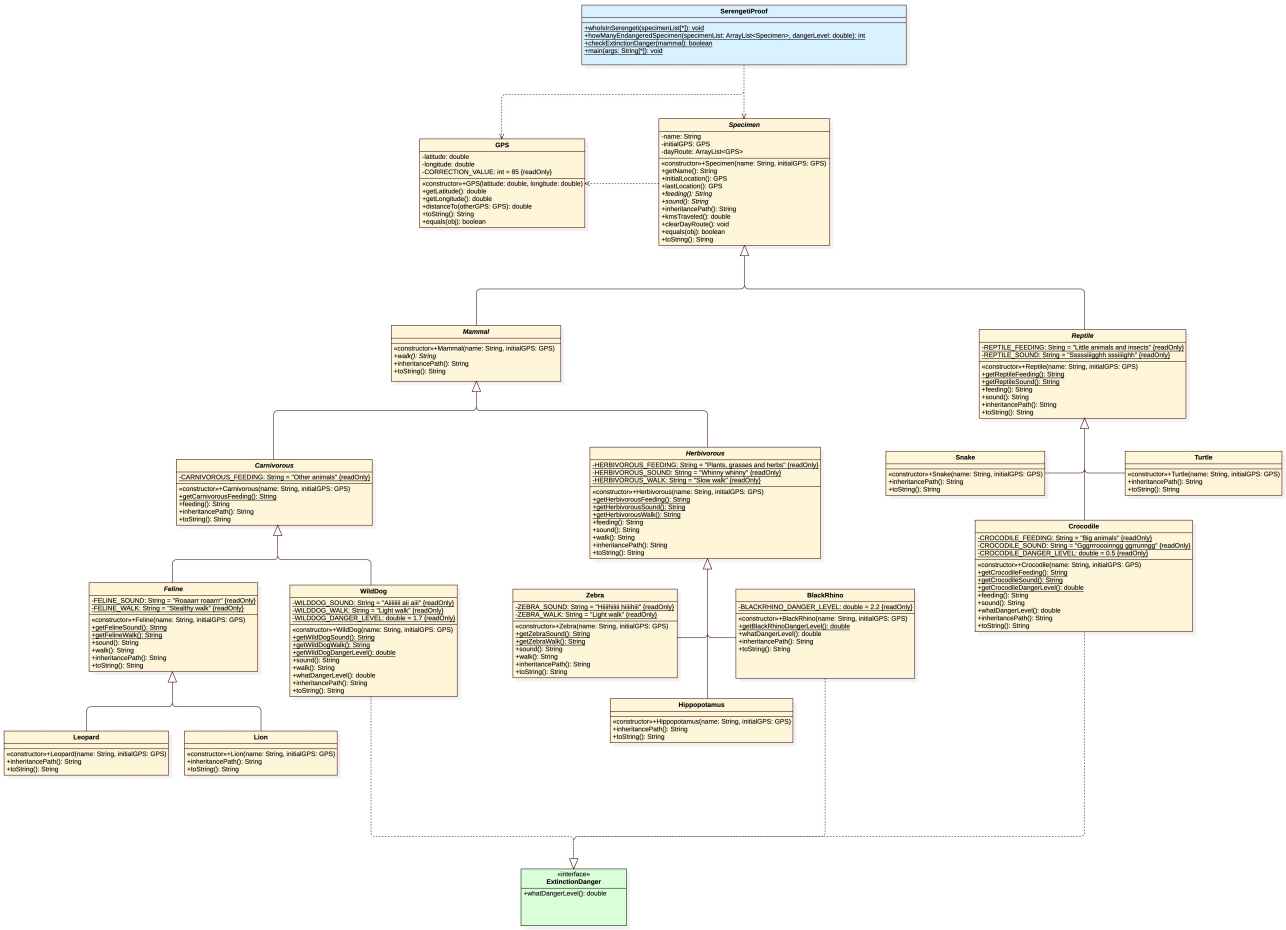
3.5 Simulating

To finish with this part of the project we have created a SeregetiProof class. The purpose of this class is to code some static methods that will help us to check how many Specimens are in the Serengeti Natural Park of each type and the characteristics of each; or check the extinction danger of some of them. Later we are going to create an ArrayList with different types of Specimens and test the methods. Due to some tasks assigned to check what we have learnt I have printed some notes such as *Runtime error* or *Compilation error* indicating that the instruction can not be done (there is more information in the code).
Here is one of the simulations done:

```
Specimens that are in the Serengeti Natural Park:
BlackRhino
- Amount: 2
- Feeding: Plants, grasses and herbs
- Sound: Whinny whinny
- Walk: Slow walk
- Danger level: 2.2
Crocodile
- Amount: 2
- Feeding: Big animals
- Sound: Gggrrrroooinngg ggrrunngg
- Danger level: 0.5
Hippopotamus
- Amount: 2
- Feeding: Plants, grasses and herbs
- Sound: Whinny whinny
- Walk: Slow walk
Leopard
- Amount: 2
- Feeding: Other animals
- Sound: Roaaarr roaarrr
- Walk: Stealthy walk
Lion
- Amount: 2
- Feeding: Other animals
- Sound: Roaaarr roaarrr
- Walk: Stealthy walk
Snake
- Amount: 2
- Feeding: Little animals and insects
- Sound: Sssssiiigghh sssiiighh
Turtle
- Amount: 2
- Feeding: Little animals and insects
- Sound: Sssssiiigghh sssiiighh
WildDog
- Amount: 2
- Feeding: Other animals
- Sound: Aiiiiiii aii aiii
- Walk: Light walk
- Danger level: 1.7
Zebra
- Amount: 2
- Feeding: Plants, grasses and herbs
- Sound: Hiiiihiiii hiiiihiii
- Walk: Light walk

Specimens in danger of extinction equal to or greater than level 1.0:
- BlackRhino: BlackRhinoExample1
- WildDog: WildDogExample1
- BlackRhino: BlackRhinoExample2
- WildDog: WildDogExample2+
There are 4 specimens with equal or or higher level of extinction than 1.0.

The variable z has Zebra as static type and Zebra as dynamic type. Is it in danger?? false

The variable m has Mammal as static type and WildDog as dynamic type. Is it in danger?? true

The variable s has Specimen as static type and WildDog as dynamic type. Is it in danger?? true
The variable o has Object as static type and BlackRhino as dynamic type. Is it in danger?? true

The variable r has Reptile as static type and Snake as dynamic type.
Compilation error: You can't cast that dynamic type into Mammal.


Process finished with exit code 0
```

3.6 Project design

3.6.1 Classes design



4. Conclusion

In my opinion, this project has been very interesting and has helped me a lot to practice the concepts learnt at the subject and improve my programming skills. Both parts are very complete and different one to another. In general, all the project works correctly and I have not had any problem.

In the first part, I have learnt a lot with the use of some basic concepts, such as encapsulation, singleton, ArrayLists, Exceptions... I have developed some algorithms that were more difficult that the ones that we have made before. The first part of the project was exigent and I worked a lot on the algorithms and simulation; the project structure was easier.

In the second part, I have learnt a lot of code re utilization, hierarchy and inheritance. In my opinion, this part has been easier than the first one, we have to think less on algorithms and more on the project structure and code re utilization because we were working with a large number of classes.

To finish with the project, I think that programming is very interesting and this subject has been very important in my academic formation.