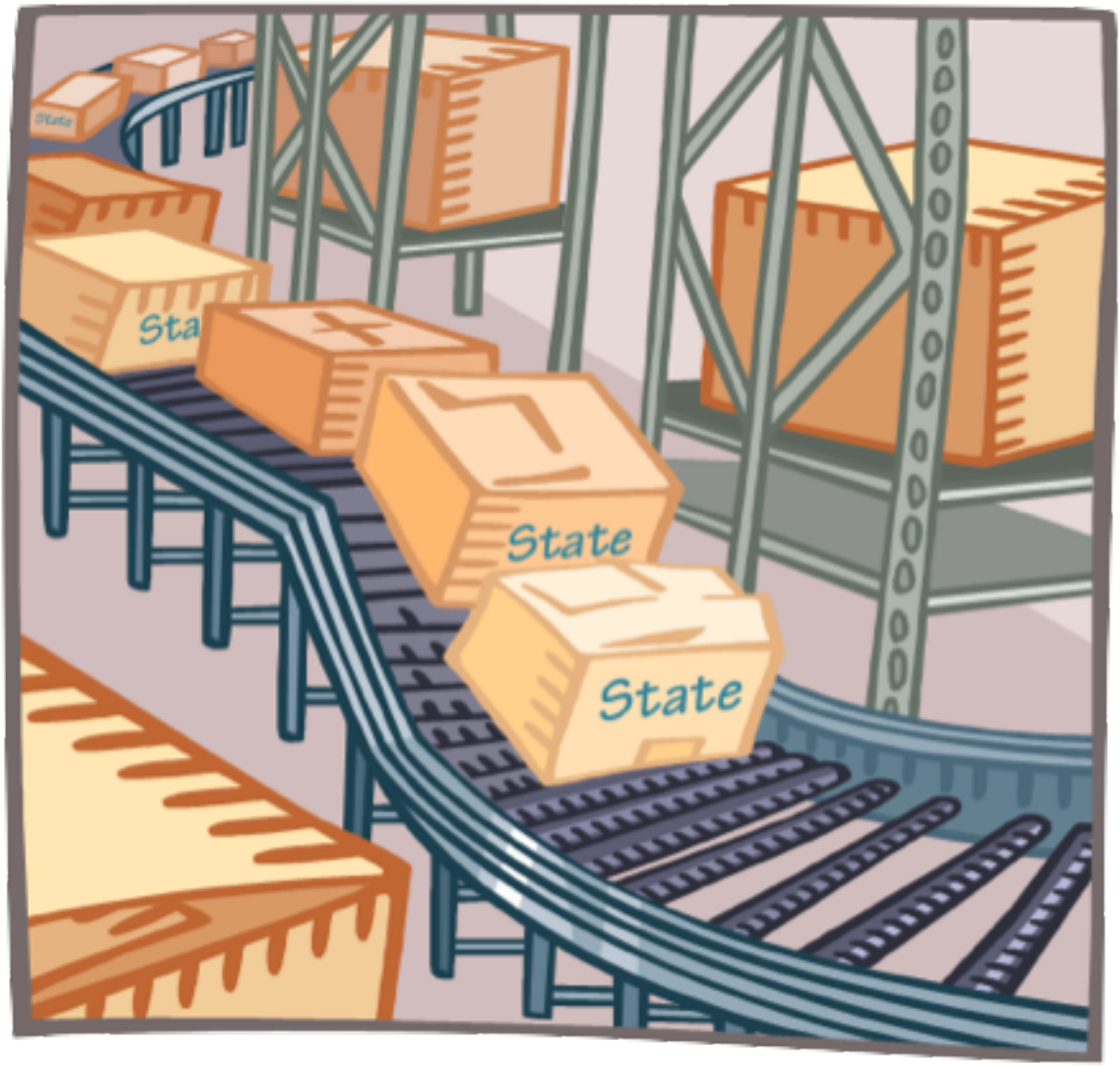
FACTORY PATTERN



Adhraa Mussalami
Arthur Issarni

Table Of Content

Introduction

Factory pattern is one of the most used pattern in Java. It is a good way to create different objects using a common interface.

With the factory pattern we use a Factory class to create an object implementing the common interface. We can use different Factories to build different set of object.

The following UML diagram shows a simple use of the factory pattern. We can see the common interface “Shape” an the different objects implementing this interface. The FactoryPatternDemo class will use ShapeFactory to get the type of object needed.

Generic Game

For this assignment we decided to make a simple game. This game is a 2D game where you can move, jump and use your tool (gadget) on the evil target to win.

This game contains multiple objects (character, girl character, evil character and a gadget). In order to play the game in two different versions (Minion and Mario) we used the Factory Pattern.

Application of the factory pattern

The Factory Pattern allows us to have different kind of objects from an interface. Then we can create the objects we need by using a Factory.

In our case we have 4 different interface : figure, gadget, target, female. If we want to play the minion version, minionFactory class will create each object of the respective type.

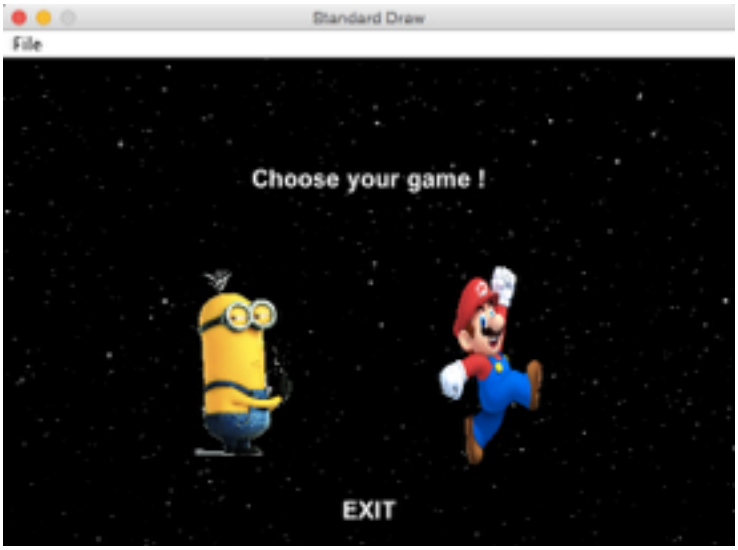
How to run the program

To run the program you can import the project into Netbeans or run it by executing the .jar file using the following command.

```
java -jar path/to/the/jar/file
```

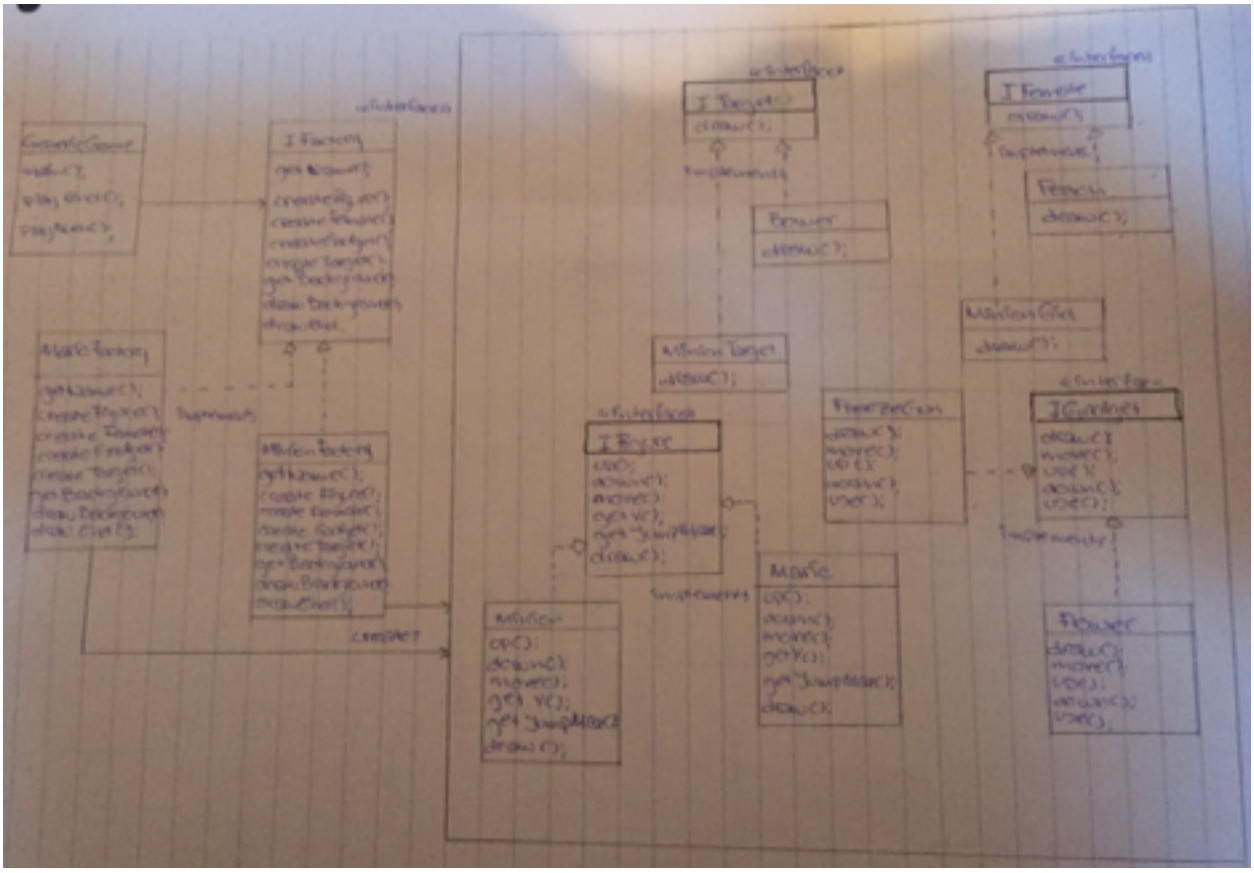
Life cycle

By running the game you will enter the menu screen where you can choose the version you want to play.

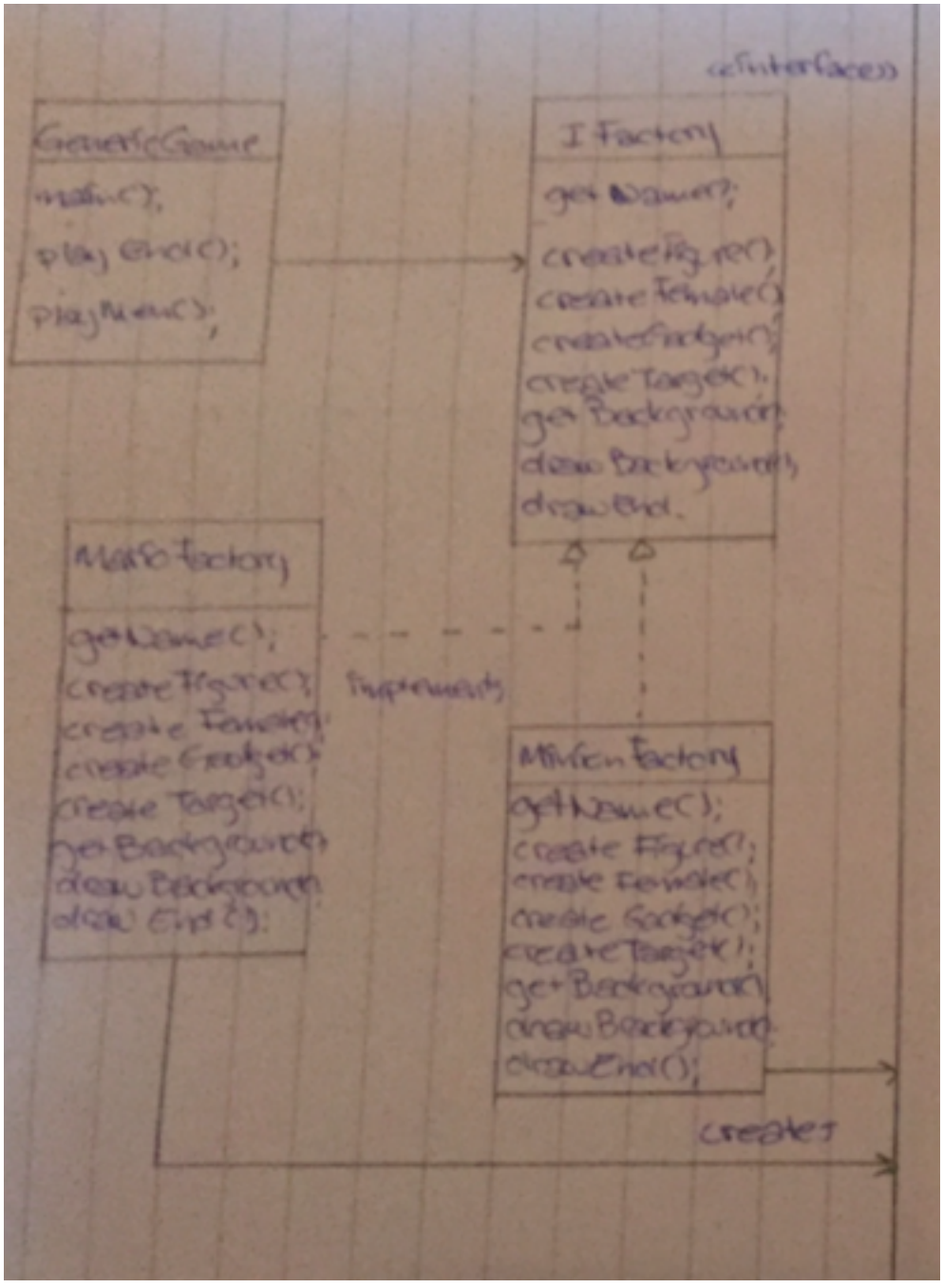


During the game, you control your character by using the left, up, right arrow keys and space bar to use your tool. When you hit the target, you win and you can start a new game by going back to the menu.

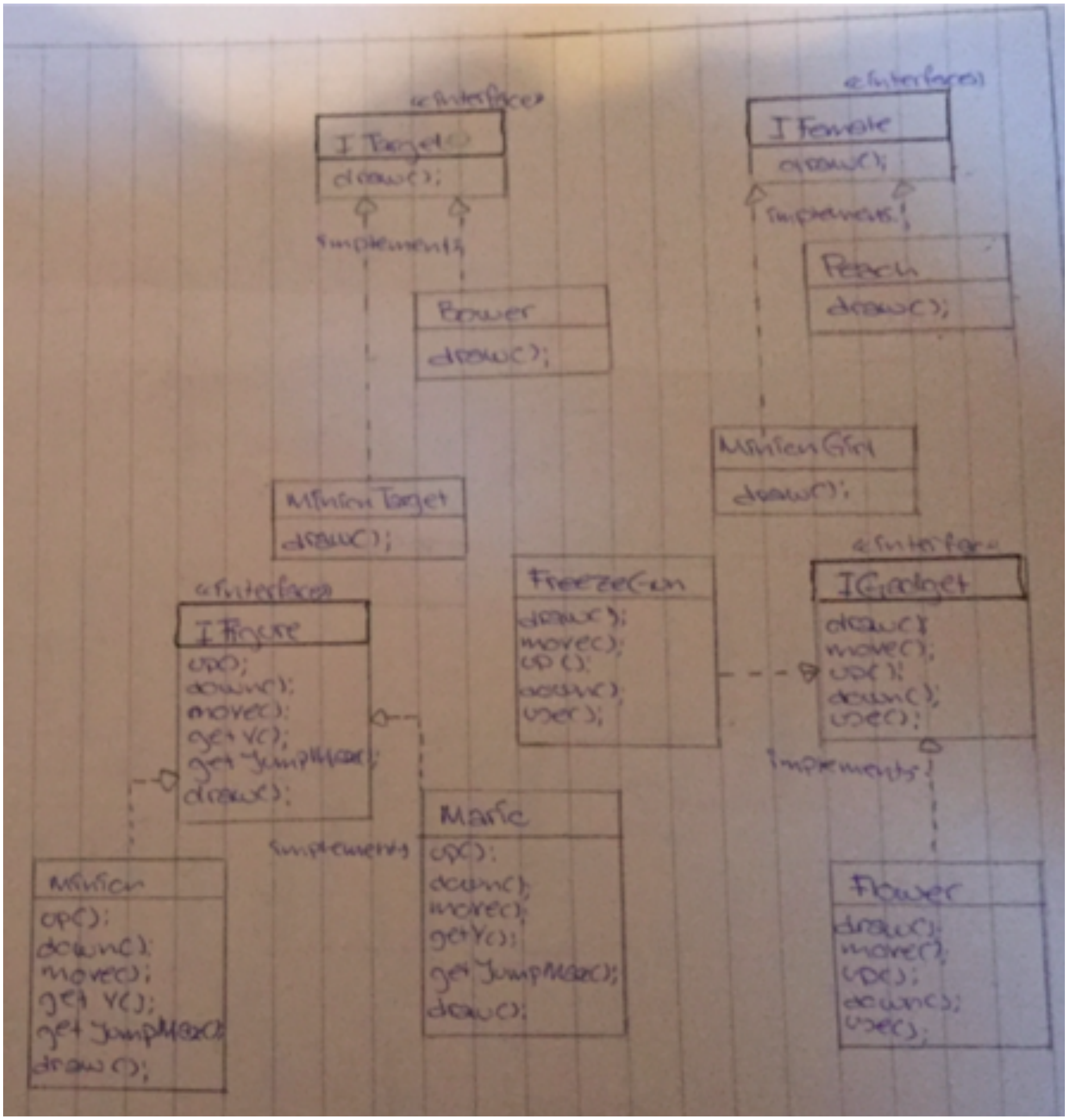
UML Diagram



Left Side



Right Side


«interface»
ITarget
draw();
implements
Bower
draw();
«interface»
IFemale
draw();
implements
Peach
draw();
MinionGirl
draw();
MinionTarget
draw();
FreezeGun
draw();
move();
up();
down();
use();
«interface»
IGadget
draw();
move();
up();
down();
use();
implements
«interface»
IFigure
up();
down();
move();
getY();
getJumpMax();
draw();
implements
Mario
up();
down();
move();
getY();
getJumpMax();
draw();
Minion
up();
down();
move();
getY();
getJumpMax();
draw();
Flower
draw();
move();
up();
down();
use();