# Basic Programming: Final Project

First a disclaimer. Unfortunately I severely underestimated the time I would need to implement all my ideas. The game is therefore not entirely finished. Game aspects, that I had taken into account to include (meaning I had written all other parts of the code in the idea of eventually being able to include them), but which I haven’t yet finished, will be written *in red and cursive text*.

## The Backstory and Rules

### The backstory

You have heard stories about undiscovered lands, with fertile soils, ores and gold fields. With the help of Captain Rick, Jack and Judy, you will embark on a long journey towards these undiscovered territories, where you will set up your first settlement which you will lead and grow to a thriving city. Soon more settlements will be established and you will be emperor over the new world! Riches and adventure awaits!

### The inspiration

The inspiration for this game came from one of my favourite childhood games: Colonists of Catan, the card game for two[[1]](#footnote-1). I have spent hours playing the game and mastering the tactics to win. Now, after the board game for 4, and the card game for 2, I propose the computer game for 1.

### The game

#### The essence

The game is made as a conversation between 4 characters; captain Rick, Jack, Judy and you. As of the start of the game conversations between these characters, introduces the player to the game and helps the player set up the game. The player has the option to choose a random map, to specify the width and height of the map and even a more advanced setting of specifying individual tile types. After the set-up of the game is done, the player can start building their empire. Throughout this process the player is left slightly more to themselves, and is repeatedly prompted with the question: “What would you like to do next?”. The player thus goes through the game by answering the questions he is asked and giving commands. A list of valid commands is given when the player types ‘help’.

#### Some basic game play aspects and rules

For the player who knows the card game, many aspects will be recognisable, though implemented slightly differently.

##### The Game Map and Resources

As described previously, the player can set several aspects of the playing field of game map. Both the size and the tiles can be set as wished, or generated at random. The game map will form the bases of the empire, it is the new world you have discovered after your journey overseas.

The game map consists of several different land tiles, each holding a different resource. Similarly to the card game, 6 different types of resources are spread amongst the world:

1. Fields can yield Grain
2. Mountains can yield Ore
3. Pastures with sheep yield Wool
4. Forests yield lumber
5. Hills with clay yield bricks
6. Gold Fields yield gold

Each of these tiles also have a ‘yield’ attribute, specifying the amount of the resource is yielded at each turn.

##### The Empire

There are several levels of game play. First there is you, the emperor ruling over an empire. The empire starts off with a certain amount of resources, with which you can build settlements. Each settlement within your empire can be seen as a fairly independent part of the game. It is your task to rule over all the different settlements, which all have their own resources and population. It is thus very well possible that within your empire there are some flourishing cities and also some decaying settlements. However, *by redistributing the resources wisely*, you can build your empire to be the greatest the world has ever seen!

**Summary:** **An Empire has its own resources and consists of a list of Settlements.**

##### Settlements

Settlements are the basis of civilizations. **They yield resources, have a population and can contain buildings.**

Each settlement must be built on 4 uninterrupted neighbouring tiles, none of which are mountains and none of which have other buildings already built on them. Each settlement starts off with a territory range of 1 tile. This means that the inhabitants of the settlement will be yielding the resources within a 1 tile radius of the settlement and buildings can be built only within this territory. Imagine the settlement is depicted in black. The settlement territory is then depicted in yellow. Similarly to the card game, a settlement can be upgraded to a city, provided an additional cost. At this point the radius becomes two tiles. The orange shading shows the additional territory of a city.

Note that tiles can only belong to 1 settlement and will belong to the first settlement they belonged to. This is illustrated in the left figure in which a new settlement is built.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

Each settlement forms a rather independent whole. This means that each settlement has its own resources, with which it can buy buildings to build within its own territory. Both the population and the buildings cost resources in food (Grain) and maintenance respectively. At the end of each turn, the yields and costs are added and deducted from the resources, and when the settlement cannot pay its maintenance, the buildings start to decay. A settlement in decay will be less attractive for its population. When a settlement needs money, it can get a loan, which can be specified for each of the 6 resources and a payback plan can be set up.

**Summary**: **Each settlement has its own resources, can contain buildings and has a population.**

##### The population

Each settlement has a population. *The yields of the settlement will depend on the number of people that can work the fields and mine the resources.* It is thus of vital importance to keep the population of the settlement happy. Some aspects that will determine the (un)happiness of the citizens are the number of unemployed people, the poverty index (indicating the lack of grain to feed the citizens), and the decay of the settlement infrastructure.

##### Buildings

Unfortunately I have not yet specified any buildings. An abstract indicating some aspects of what all buildings have in common is made. An elaborate implementation of this abstract template for buildings is the previously discussed ‘Settlement’, which is a special kind of building, that does not require to be built in a settlement.

*All other buildings can only be built in settlements. And as in the card game there would be ‘settlement buildings’ and ‘city buildings’. The former can be built in both in settlements and in cities, and the latter can only be built in cities.* Though this has not been implemented yet, I have already included some methods and variables that were supposed to allow for this implementations.

As indicated in the abstract class ‘Building’, all buildings will have a name, mapSymbol, buildinCost, maintenanceCost, buildingDate and employmentOpportunity. *Additionally most buildings would have specific roles in the game. Tile improvements such as windmills, mines, lumberyards and farms would increase the yield of a tile, while a mint would allow money and thus the trade of gold for resources, etc.*

## Description of the classes

### Package ArrayMaths

A package mainly made to do maths with the resources: double[] resources = new resources[6]

#### ResourceCalculator

Holds methods to add resources, deduct resources and check whether enough resources are available.

#### ValidCostChecker

Checks whether all resources are positive. (It might have been more logical to include this in the ResourceCalculater class instead of making it a separate class.)

### Package Building

#### Abstract Building

Buildings have been discussed earlier. As mentioned before, I have only finished the abstract class, upon which more building should be based. Aside from the most important variables that each building should hold, such as mapSymbol, buildinCost, maintenanceCost, buildingDate and employmentOpportunity, the abstract contains many methods, which are mostly used for verifying whether a building may be built.

Method checkTiles: Evaluates the tiles for forming one whole, not having mountains and not having other buildings.

Method areNeighbours: Evaluates whether the tiles form a whole.

Method areDistinct: Evaluates whether distinct tiles were given.

Method inCity: Evaluates whether all tiles are in a city. (Needed for city buildings.)

Method calculateDeforestationCost: adds costs for each tile that is a forest.

Method getTfromR: gets the tiles from their reference numbers.

Method setAssociatedBuildingOfTiles: associates the new building to the tiles it is built on. (=> building refers to tiles and tiles refer back to building.)

#### Settlement extends Building

A lot is already explained previously. Aside from the needed variables and the references to the tiles, territory, buildings and empire, the most important feature of the settlement is that the constructor is set to private. And a ‘construction site’ the static method ‘build’ was created to ensure that a Settlement object is only created after thorough evaluation of the validity and compliance to the rules of the game. This should be done for all the buildings. The settlement class is an elaborate example of how all other, more simple building subclasses should be made!

The settlement class also has a lot of methods concerning the resources, yield, maintenance etc. calculations that need to be done at the end of every turn.

### Package Catan

Catan holds a set of classes that do not clearly belong to the same type, but all have to do with the game play.

#### Empire

As described above an empire has resources, an associated world, and a list of settlements. Aside from setters, and getters, it also has some methods to allow for getting information on the empire.

#### Game

In the end I did not use this class. However the original idea behind it was that if I would ever wish to extend the game to a multi-player game or a game in which you play agains the computer, it would be possible to have several empires playing in one world. This class thus has a world and a list of empires.

#### Interface Inspectable

Since it is important to inform the player, the interface inspectable was made. Many classes implement this interface, forcing them to have a method ‘inspect’. Building, Empire, Tile and Population all implement this interface.

#### Interface

This contains the main method and the text based interface with the player. It starts with a set-up and game initialization and then continues in a loop of turns until the player requests an end of game.

#### YearEnd

This method groups together all the methods that need to be implemented at the end of a turn.

### Population

#### Population

A class dealing with all the population variables and methods that increase and decrease the population size.

### World

#### ResourceNameGenerator

Give it an integer and it will return the name of the resource. (Just an easy way to keep track.)

#### Tile

6 different tiles exist, each with a different resource. Tiles also have a yield, can lie within the settlement of a territory and can have a building on them.

#### World

The world or game map contains a list of tiles. These can be generated at random or specified by the player in various ways.

## Relationships between the classes

### Object oriented aspects

The abstract class Building should form the bases of all buildings. The class Settlement is an example of that.

The interface Inspectable forces all Building, Tile, Empire and Population to have a method called ‘inspect’ that prints out information to the player.

### Other relationships

The world has tiles each tile can be associated to a settlement and can have a building on it.

Each building (aside from settlements themselves) is part of a settlement and is built on tiles.

Each population is part of a settlement.

Each settlement can have buildings and has a territory of tiles and an associated *empire (for multi-empire game implementations)*.

An Empire has an associated world and has a list of settlements.

It is thus clear that the different objects refer to each other.

## Strengths and weaknesses of the project

### Strengths

1. Well thought through planning. I have put a lot of thought before starting the project in the planning. I made sure to make the game as versatile as possible and included many possibilities to further develop the game. (E.g. many more buildings, possibility of multi-empire game, etc.)
2. I tried to include many ‘checks’ to evaluate the input to methods.
3. A lot of potential for a fun game. Though the game is not finished, I have tried to think of a fun game with many special aspects. The fact that an empire contains several settlements, which all work relatively independently, could, I think make for an interesting and challenging game.

### Weaknesses

1. **The main weakness is that the game is not finished**. I have wrongly estimated the time it would take to implement my ideas and had not planned out enough time to finish the task at hand. Though the implementation of the buildings has been made easy by a well-planned abstracted method, the game is not playable at the moment due to the lack of any buildings.
2. **The text based Interface:** Again due to running out of time, the interface is lacking in many ways. There are a lot of glitches and efforts to try and catch input exceptions (mainly mismatch exceptions) failed.
3. **Mistakes remain:** While testing the game some mistakes remain. The setting of specified tiles works when directly implementing the World constructor method with an array of integers, but something goes wrong in the interface class. I have also noticed some warning messages I had set myself when they should not appear.

**The code is clearly not finished and many logical errors seem to remain.**

1. **Handling exceptions:** Though I have tried to handle exceptions in my code, it is clear that I have not fully gotten it down. I tried to do this in the interface, but had to comment out the code I wrote, because I did not manage to make it work correctly.

1. http://www.catan.com/game/catan-card-game [↑](#footnote-ref-1)