CE0825a - Object Oriented Programming II
4: Map Case Study

James A Sutherland

Abertay University

Monday, 1st February 2016

Map Example Introduction

First chapter describes the simplest possible map: a grid of yes/no squares.
Roughly what Robby the Robot roamed in 1st year: the whole world is one boolean[][].

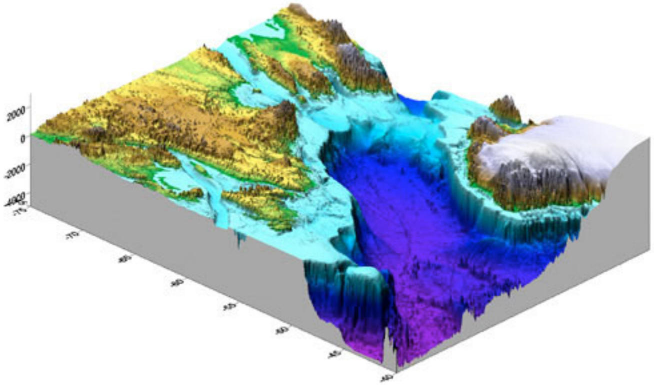
More complex maps

That's the simplest possible map: a totally flat grid of squares.
You can improve that:

- More/smaller cells
- Add height information for basic 3d
- Add colouring or texture information
- Interactivity – cells can be deadly, useful...

Map Example 1

A more impressive mapping example[1]



[1]Image courtesy of Igor Yashayaev, Bedford Institute of Oceanography, Fisheries and Oceans, Canada.

Building that in Java

Chapter 2 presents some simple example code – *not* an example of good Java (that comes later), but useful bits.

For those joining directly, in 1st year we programmed a simple robot driving around a map like this: turn left/right, check for obstacles in front, move forward.

New to Java? Read through it all carefully, try answering the practice questions to get the hang of it.

Already taken 1st year Java, but need a reminder? Useful revision to read through and make sure you understand it.

Java language

Three key things Chapter 3 introduces:

- Abstractions: making modular components
- The ‘final’ keyword for classes/methods/fields not to modify
- Enum types

Enumeration types

More maintainable, better error checking.
So you use 1=grass, 2=road, 3=river.
Then you need to add another colour of grass ... or a puddle, or tree ... or just maintain the code later.
Much easier all round to have GRASS, ROAD and RIVER, isn't it?
Also checks all your switch statements cover all the options: add one, get warned if you miss anything.

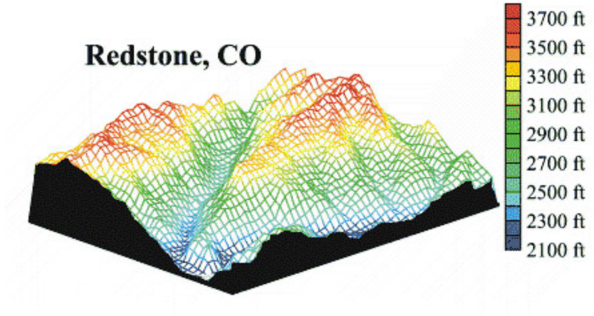
Java final modifier

Generally, indicates “this thing should not be modified”.
On a field: makes it either constant, or set in stone by the constructor. (Fail to set in the constructor, get told off.)
On a method: you can’t hide or override in a subclass.
On a class: you can’t subclass (extend) it.

Next Steps

You could use an Enum to make your map more interesting. For now, let's try an integer to reflect the height of each point. Then you can colour heights according to a spectrum, and get quite a nice map.
(You can plot these directly from published OS map contour lines, too.)

Map Example 2



Remaining Chapters – Future Work

By the end of chapter 3, we have multiple `TerrainType` values, with a flag `passable` for each.
Later chapters introduce things we'll look at more closely later:

- an ITiledMap interface
- Random numbers
- Strategy patterns
- Disabled accessibility, alternate UIs
- Audio example
- Rendering to JPanel (Swing) with scaling
- Jar files

Week 4 Lab Tasks

Work through the first **three** chapters online.
Extent the code provided (eg3) from booleans to an Enum with at least three different options.
Add code to print the map out on the console as “ASCII art” – tip: Java isn't limited to true ASCII, you can use Unicode symbols, like 2588 (FULL BLOCK) and 2591-3 (LIGHT, MEDIUM, DARK SHADE).
There is even a CHRISTMAS TREE character! Check for support online.[2]

[2] `http://www.fileformat.info/info/unicode/font/fontlist.htm?text=%F0%9F%8E%84+-+CHRISTMAS+TREE+%28U%2B1F384%29`