Introduction to Java

The plan

- Racket will return!
  - Final project will be writing a Racket interpreter *in Java*.
- Lecture will not recount every single feature of Java.
  - You may need to do some digging on your own.
  - Lots of help online (Google is your friend).

Java Resources

- Java tutorial
  - http://docs.oracle.com/javase/tutorial/java
- Java documentation
  - http://docs.oracle.com/javase/6/docs/api
- And if you're confused about anything, Google will find it.
  - There's so much Java stuff on the web because most undergraduate curriculums now teach Java as their first or second language.

Logistics

- We will use Java version 6.
  - Java 7 is compatible, but not as widely adopted.
  - (It also won't run on my Mac, so we're using v6).
- Many powerful IDEs out there.
  - I will be using an IDE called NetBeans, which is free.
  - Installation instructions will be on the class webpage.

Next Assignments

- Project 4 – out now, still in Racket
- Project 5 – Probably a Java warmup assignment, given after Easter break.
- Project 6 – Still thinking about it, probably given out 2nd week of April.
- Project 7 – Probably the Racket interpreter in Java. Will be due near the end of classes.

History of Java

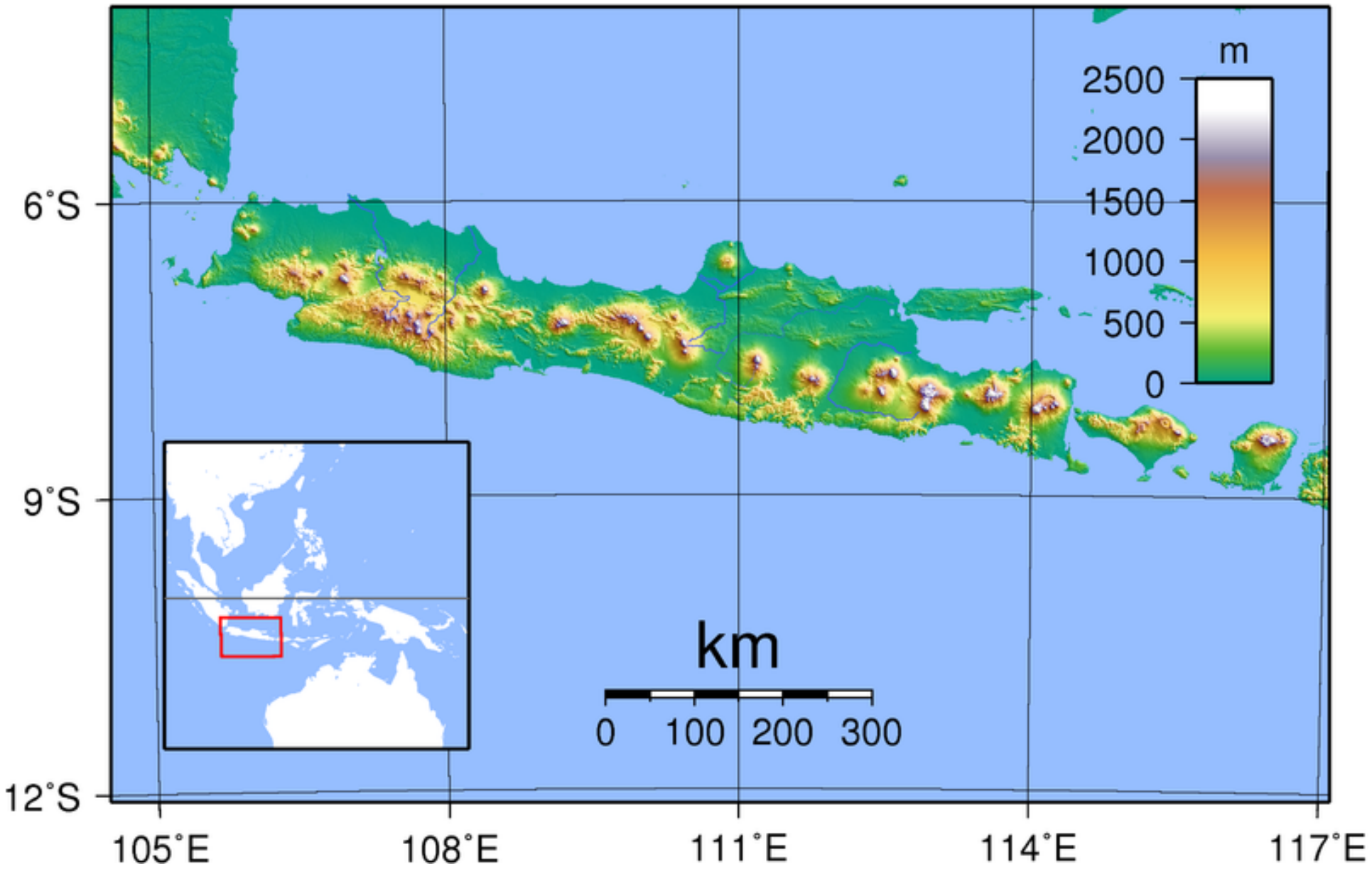
History of Java

- Java was first used in the 15th century, in Yemen, and quickly spread to Egypt and North Africa.

The Real History of Java

The Real History of Java

- Java is millions of years old and 135 million people are see Java every day.


m
2500
2000
1500
1000
500
0
km
0
100
200
300
6°S
9°S
12°S
105°E
108°E
111°E
114°E
117°E

The Real, *Real* History of Java

- The Java project was initiated at Sun Microsystems in 1991.
  - Supposedly named after the large quantities of coffee the language designers drank.
- Originally was designed to be embedded in consumer electronic devices, like cable TV set-top boxes, but it was too advanced for the cable television industry at the time.
- Language evolved into a general-purpose programming language.

- Java was designed to use a syntax similar to C and C++.
  - Lots will be familiar.
- Java is (almost completely) object oriented.
  - All data types are classes, except for the primitives like int, long, float, double, char, boolean.
  - All code is written inside some class.
    - All functions are methods (no free-floating functions).
  - Single inheritance only (C++ allows multiple).
- Statically typed (like C++).
- Has *generics* (similar to C++ templates).

- Same basic programming properties as C++.
  - Must declare variables before use, say what type they are.
  - If/else, for, while, do-while, switch work just like C++.
- No pointers!
  - Java uses a similar idea called references, which are "safer" than pointers.
- All objects stored on the heap (using "new").
- Garbage collection
  - No explicit allocation/deallocation of memory. ☺

Defining a class

- Take a look at the Rational class.

- Create primitive variables just like in C++:
  - int x = 4;
  - float f = 3.02;
  - boolean b = true; // note lowercase
- Strings are objects, but Java lets you create them like a primitive:
  - String s = "a wonderful string";
- All other objects are created using new:
  - ClassName var = new ClassName(args);
  - Constructor automatically chosen based on data types of arguments.

- Variables declared in a class are sometimes called *fields*.
- Instance variables (or fields) have one copy of the variable per instance of the class.
- Class variables or static variables have one copy of the variable that is shared among all instances of the class.

- Functions declared in a class known as *methods*.
- Instance methods can access instance variables, and are called using C++-like syntax:
  - ClassName var = new ClassName();
  - var.name_of_method(args);
- Class variables or static variables have one copy of the variable that is shared among all instances of the class.
  - ClassName.name_of_instance_method(args);

Class/Method/Variable Visibility

- public: available everywhere
- protected: only available to self and subclasses (not used that much)
- private: only available to self

- Common to have instance variables as private and methods that are part of the class's interface as public.

- Java traditionally uses CamelCase rather than separating_with_underscores.
- variables and methods start with a lowercase letter.
- Class names start with an uppercase letter.
- "this" works just like in C++.
- All objects by default inherit from the "Object" base class.

Getting a program started

- Each class must go in its own file, which must be named ClassName.java.
- Any class can have a public static main() method, which is where the execution starts.

Packages

- Java's standard library (all the functions that the language comes with) are organized into packages
  - A hierarchical organization system.
- In Java you "import" classes from packages, whereas in C++ you "#include" files.

Collections

- Built in classes for
  - Lists (ArrayList, LinkedList, …)
  - Sets (HashSet, …)
  - Maps (what Java calls hash tables) (HashMap)
- All of these are parameterized with generics.
  - List<Integer> intlist = new List<Integer>();
  - intlist.add(17);
  - System.out.println(intlist); // prints [17]