Unit 1: Primitive Types
Basic Java Syntax

Adapted from:

1) Building Java Programs: A Back to Basics Approach
by Stuart Reges and Marty Stepp

2) Runestone CSAwesome Curriculum

Textbook Reference

Online Textbook Think Java - 2nd Edition by Allen Downey and Chris Mayfield

For this lecture use Chapter 1

Java

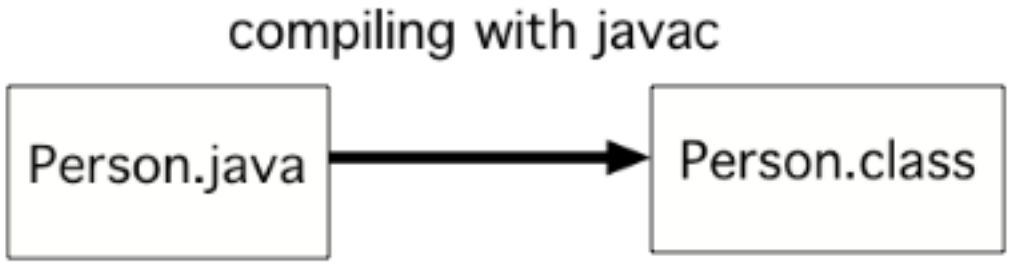
What do Minecraft, Android phones, and Netflix have in common? They're all programmed in Java!

Many of the apps you use in an Android phone or tablet are also written in Java. Netflix uses Java for some of its software too. Java is used worldwide to create software that we all use.

Java

Java is a **programming language**, which means that we can use Java to tell a computer what to do.

Computers don't actually speak Java so we have to **compile** (translate) Java source files (they end in .java) into class files (they end in .class).



The source file is something humans can read and edit, and the class file is code that a computer can understand and can run.

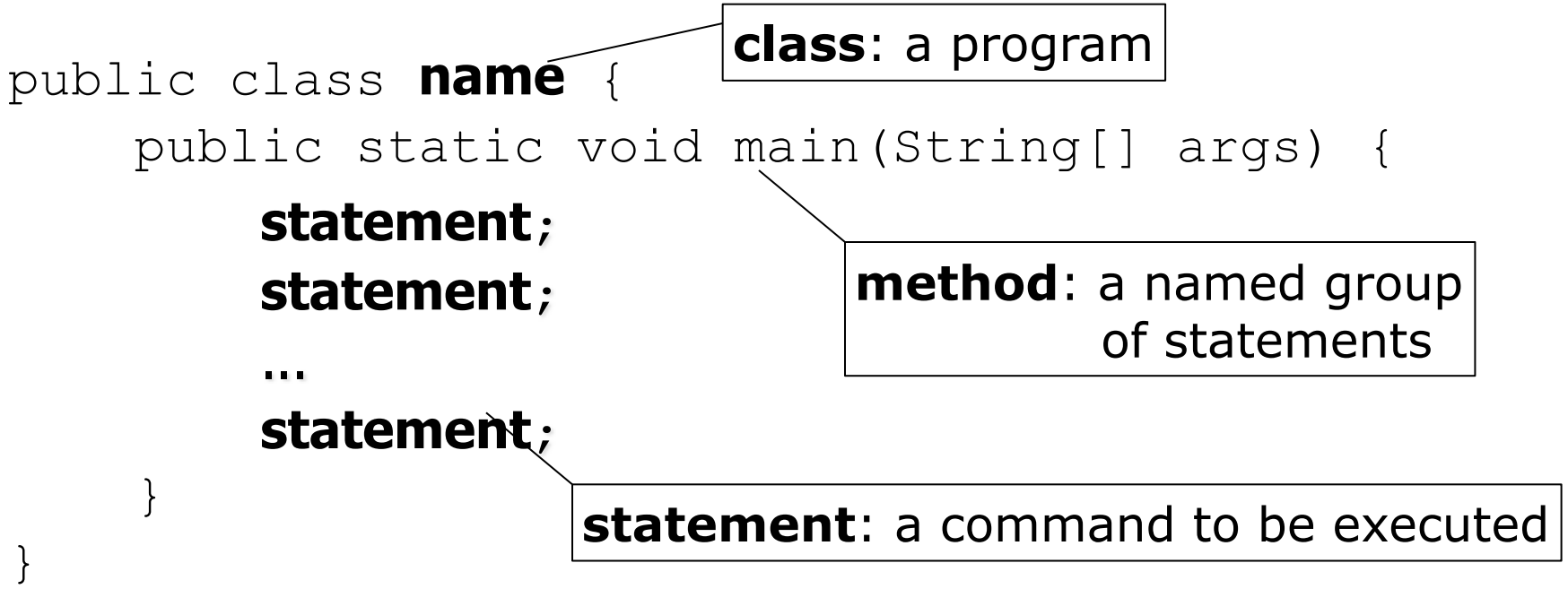
Java Terminology

- **class**:
  (a) A module or program that can contain executable code.
  (b) A description of a type of objects. (Animal class, Human class, Employee class, Car class)

- **statement**: An executable piece of code that represents a complete command to the computer.
  - every basic Java statement ends with a semicolon `;`

- **method**: A named sequence of statements that can be executed together to perform a particular action or computation.

Structure of a Java program

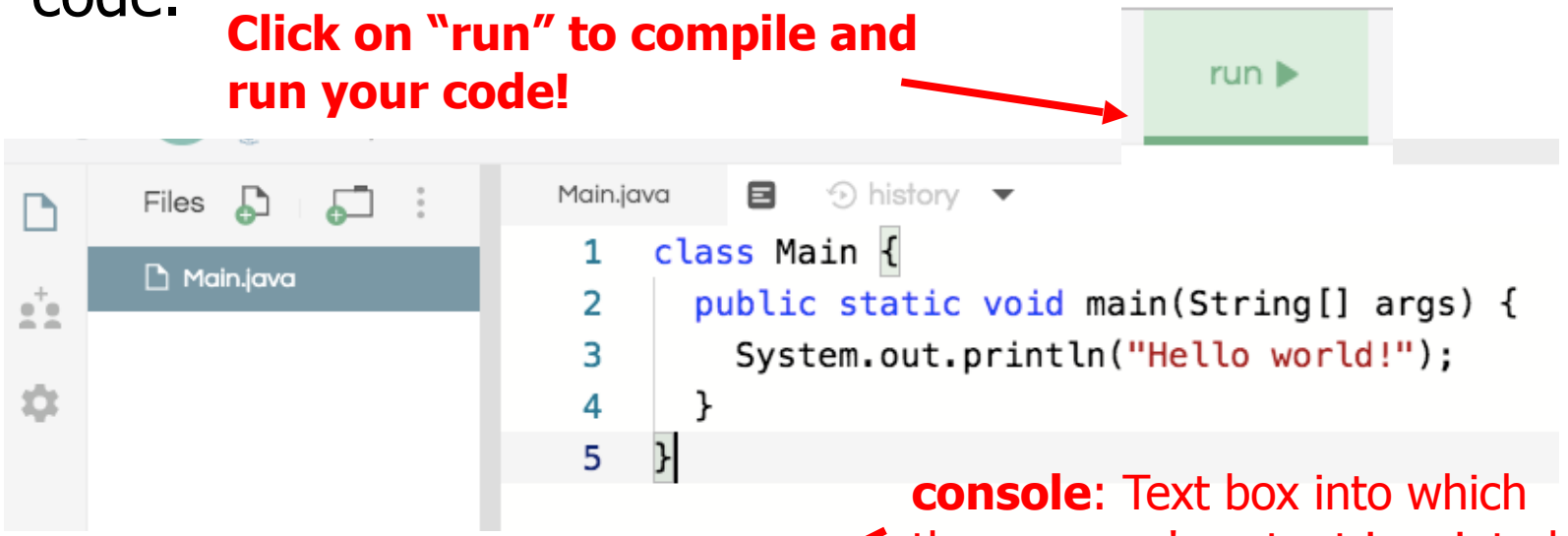


- Every executable Java program consists of a **class**, called the **driver class**,
  - that contains a **method** named `main`,
    - that contains the **statements** (commands) to be executed.

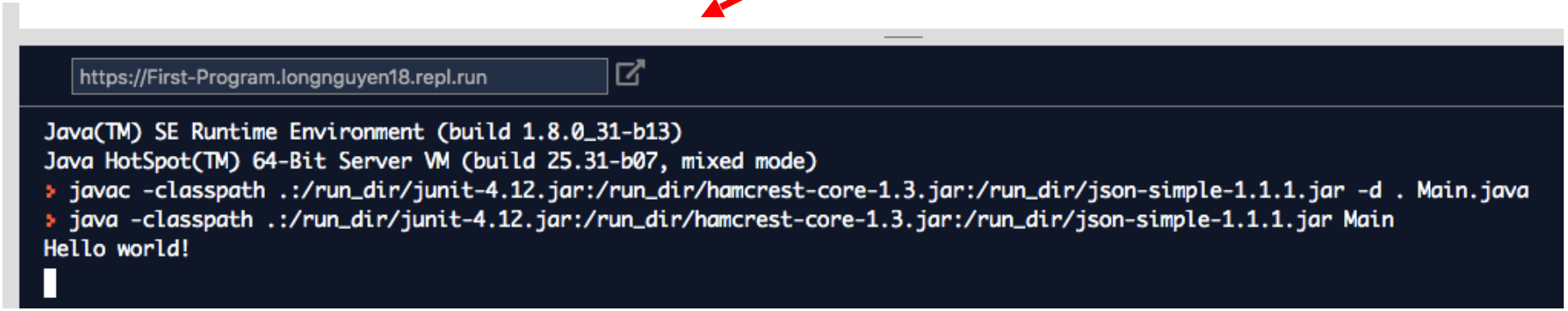
First Program: trinket.io

We will use trinket.io, an online integrated development environment(IDE), for the first part of this course to write all of our code.

Click on "run" to compile and run your code!



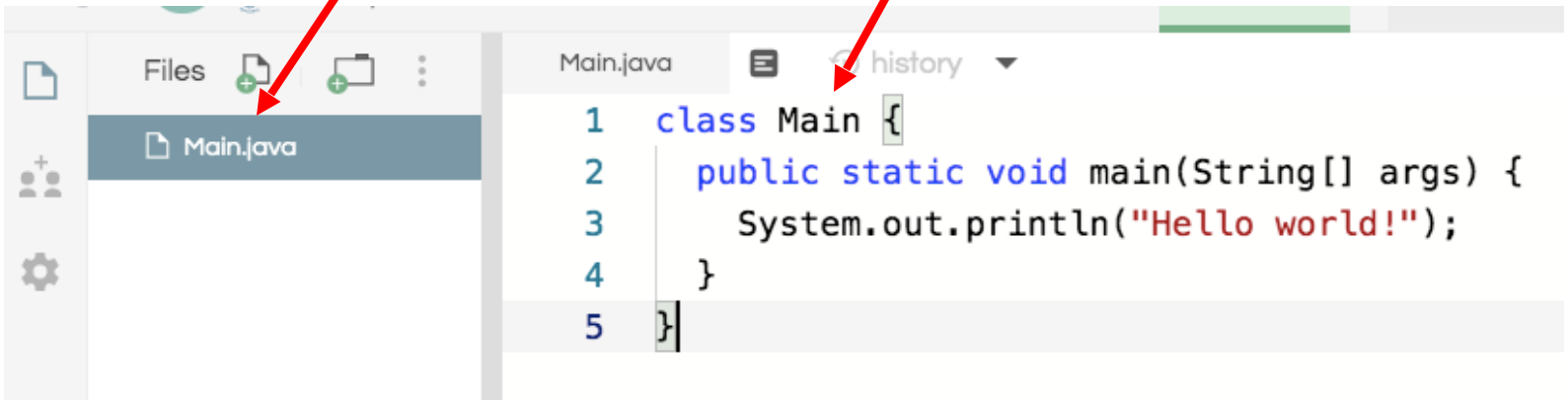
console: Text box into which the program's output is printed.



File naming

The name of the class has to match up with the name of the file.

For example, the class below is called `Main` therefore the name of the file is Main.java.



Printing

Two ways to print a line of output on the console: `System.out.println()` and `System.out.print()`.

`System.out.println()` is just the way that you ask Java to print out the value of something followed by a new line (ln).

`System.out.print()` without the ln will print out something without advancing to the next new line.

System.out.println

```
public class Welcome{
    public static void main(String[] args){
        System.out.println("Hi there!");
        System.out.println(”Welcome to CPJava!”);
    }
}
Output:
Hi There!
Welcome to CPJava!
```

The “System” in System.out.println() must be capitalized. And the command line must end with a semicolon (;).

System.out.print

```
public class SecondClass{
   public static void main(String[] args){
      System.out.print("Hi there!");
      System.out.println(”Welcome to CPJava!");
      System.out.print(”We will learn Java!");
   }
}
Output:
Hi There!Welcome to CPJava!
We will learn Java!
```

Do you see why there are two lines of output as above?

Lab: Find the errors.

```
pooblic class Errors
   public static void main(String args){
      System.out.print("Good morning! ")
      system.out.print("Good afternoon!);
      System.Print "And good evening!";


}
```

Find the Errors Assignment on Trinket and fix it there!

Submit when done or partially done and you need help. Add a comment!

Strings

- **string**: A sequence of characters to be printed.
  - Starts and ends with a `"` quote `"` character.
    - The quotes do not appear in the output.
  - Examples:


```
"hello"
"This is a string.  It's very long!"
```

- Restrictions:
  - May not span multiple lines.

```
"This is not
a legal String."
```

  - May not contain a `"` character.

```
"This is not a "legal" String either."
```

Comments

- **comment**: A note written in source code by the programmer to describe or clarify the code.
  - Comments are not executed when your program runs.
- Syntax:

```
//  comment text, on one line
```

or,

```
/*  comment text; may span multiple lines */
```

- Examples:

```
// This is a one-line comment.

/* This is a very long
   multi-line
   comment. */
```

Using comments

- Where to place comments:
  - at the top of each file (a "comment header")
  - at the start of every method (seen later)
  - to explain complex pieces of code

- Comments are useful for:
  - Understanding larger, more complex programs.
  - Multiple programmers working together, who must understand each other's code.

Comments example

```
/* Suzy Student, CPJava
   This program prints lyrics about ... something. */

public class BaWitDaBa {
    public static void main(String[] args) {
        // first verse
        System.out.println("Bawitdaba");
        System.out.println("da bang a dang diggy diggy");
        System.out.println();

        // second verse
        System.out.println("diggy said the boogy");
        System.out.println("said up jump the boogy");
    }
}
```

Indent Nicely!

```
public class Welcome{ public static void main(String[]
args){ System.out.println("Hi there!”
);System.out.println(”Welcome to CPJava!");}}
```

The code above will compile and run correctly. Java ignore whitespaces. But it is very hard to read, please make an effort to indent nicely!

```
public class Welcome{
    public static void main(String[] args){
        System.out.println("Hi there!");
        System.out.println(”Welcome to CPJava!”);
    }
}
```

Lab: Samiam

Create a new trinket "Samiam" on your trinket.io account and write a program that has the following outputs:

You must use exactly 5 different print statements.(println and/or print).

Output:

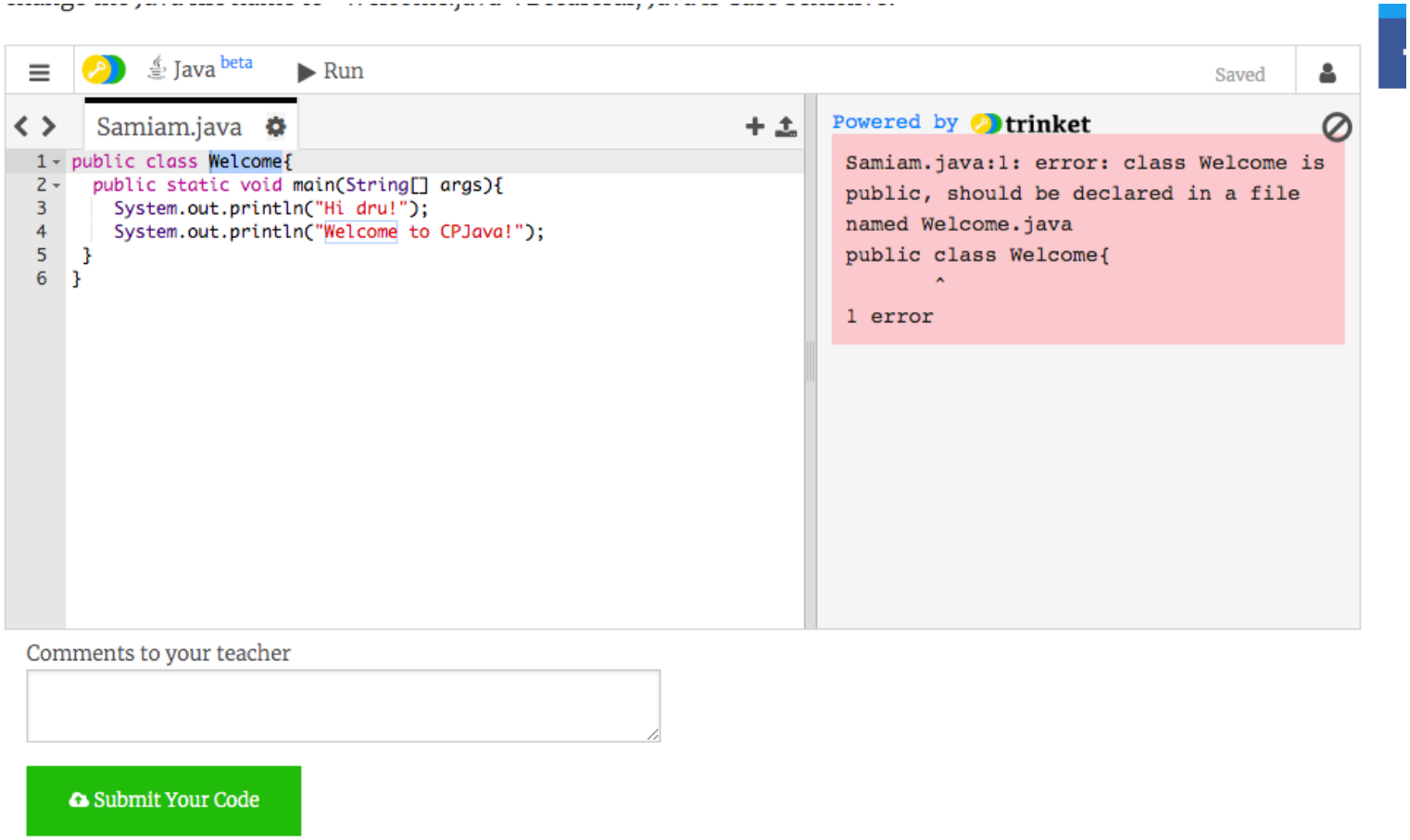
I am Sam. Sam I am. I do not like them, Sam-I-am.

I do not like green eggs and ham.

Submit your code when done. or partially done to get help!

Lab: Samiam

Submit your code when done. or partially done to get help!



References

1) CPJava Website
2) CPJava Google Classroom
3) CPJava trinket.io Classroom
4) Runestone CSAwesome BUSHSCHOOL_CPJAVA Course
5) Online Textbook Think Java - 2nd Edition by Allen Downey and Chris Mayfield
6) Building Java Programs: A Back to Basics Approach by Stuart Reges and Marty Stepp