CS 106A, Lecture 25
Life After CS 106A, Part 1

Plan for today

- Life after the ACM Libraries
- Life after Java
- Life after PC programs: Internet applications

Learning Goals

- Have fun!
- Get excited about programming after 106A!
- Be familiar with the client-server model

Plan for today

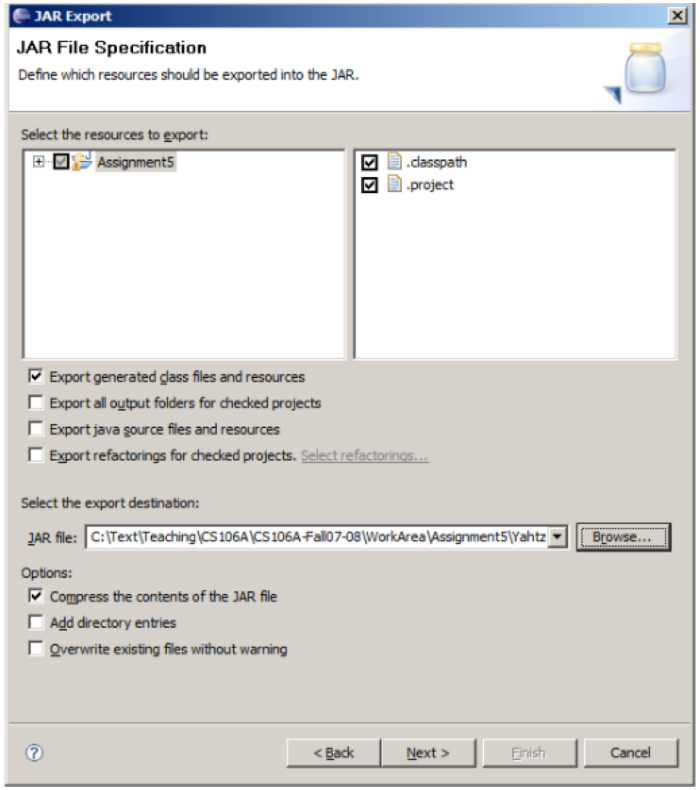
- Life after the ACM Libraries
- Life after Java
- Life after PC programs: Internet applications

Export to JAR

- **JAR**: Java Archive. A compressed binary of a Java program.
  - The typical way to **distribute a Java app as a single file**.
  - Essentially just a ZIP file with Java .class files in it.

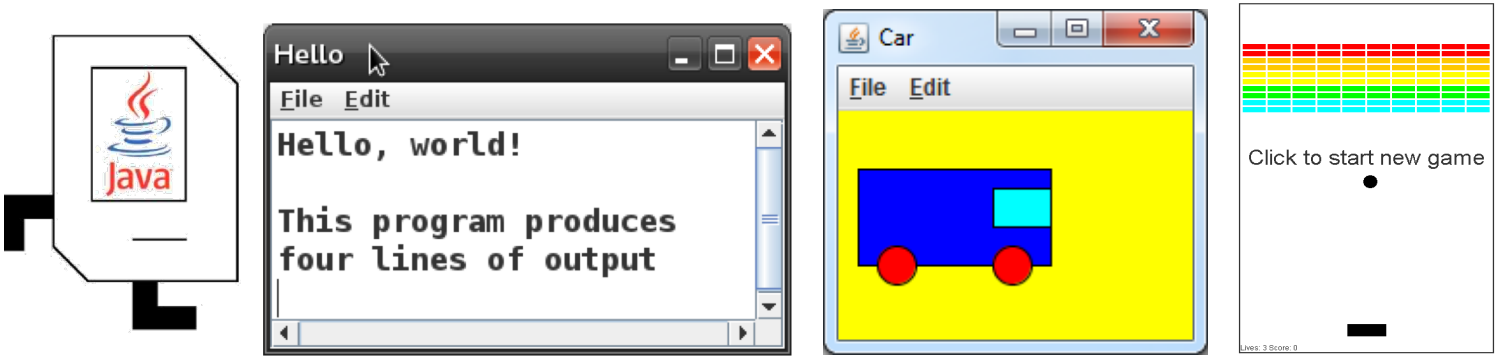
- Making a JAR of your project in Eclipse:
  - File → Export ... →
    Java → **JAR File**

- *see handout on course web site*



Life after the ACM Libraries

- All quarter we have relied on the **ACM Java libraries**.
  - Karel, ConsoleProgram, RandomGenerator
  - GraphicsProgram, GOval, GRect, GOval, GLine, GImage, ...



- Today we will see how **standard Java** programs are made.

Using the ACM Libraries

```
import acm.program.*;

public class MyProgram extends ConsoleProgram {
    public void run() {
        println("Hello, world!");
    }
}
```

- This is a console program written using the ACM libraries.
  - It uses the **ConsoleProgram** class to represent a console.
  - The **run** method contains the program code.
  - The **println** method prints output to the graphical console.

Pulling Back The Covers

```
MyProgram program = new MyProgram();
...
program.init();
...
program.run();
...
```

Pulling Back The Covers

```
public static void main(String[] args) {
    MyProgram program = new MyProgram();
    program.init();
    program.run();
}
```

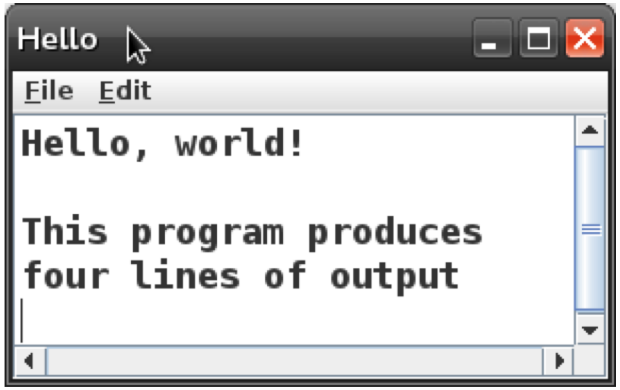
A Barebones Java Program

```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

- The method **`main`** is the true entry point for a Java program.
  - It must have the exact heading shown above.
  - The `String[]` **`args`** are "command line arguments" (ignored).
  - The way to call **`println`** in any program is `System.out.println`.
  - Java methods should be **`static`** unless instance methods of a class.

Console Programs

- What does the **`ConsoleProgram`** library class do?
  - Creates a new graphical **window**
  - Puts a scrollable **text area** into it
  - Provides `print` and `println` commands to send text **output** to that window
  - contains a **`main` method** that calls your program class's `run` method
    - `ConsoleProgram`'s `run` is empty, but you extend and override it




```
public class Hello extends ConsoleProgram {
    public void run() {
        println("Hello, world!");
    }
}
```

ACM console input

```
public class Age extends ConsoleProgram {
    public void run() {
        String name = readLine("What's your name? ");
        int age = readInt("How old are you? ");
        int years = 65 - age;
        println(name + " has " + years
                + " years until retirement!");
    }
}
```

- The ACM library has simple console input commands like `readLine`, `readInt`, `readDouble`, and so on.
- These methods display a 'prompt' message, wait for input, re-prompt if the user types a bad value, and return the input.

Java console input

```
public class Age {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.print("What's your name? ");
        String name = console.nextLine();
        System.out.print("How old are you? ");
        int age = console.nextInt();
        int years = 65 - age;
        System.out.println(name + " has " + years
                + " years until retirement!");
    }
}
```

- In standard Java, you must create a `Scanner` or similar object to read input from the console, which is also called `System.in`.
  - It does not automatically re-prompt and can crash on bad input.

Graphics Programs

The ACM library does several things to make graphics easier:

- Automatically creates and displays a **window** on the screen.
  - In standard Java, we must do this ourselves; it is called a `JFrame`.

- Sets up a **drawing canvas** in the center of the window
  In standard Java, we must create our own drawing canvas.

- Provides convenient methods to listen for mouse events.
  - In standard Java, event handling takes a bit more code to set up.

ACM GUI example

```
public class ColorFun extends Program {
    public void init() {
        JButton button1 = new JButton("Red!");
        JButton button2 = new JButton("Blue!");
        add(button1, SOUTH);
        add(button2, SOUTH);
        addActionListeners();
    }

    public void actionPerformed(ActionEvent event) {
        if (event.getActionCommand().equals("Red!")) {
            setBackground(Color.BLUE);
        } else {
            setBackground(Color.RED);
        }
    }
}
```

Java GUI example

```
public class ColorFun implements ActionListener {
    public static void main(String[] args) {
        new ColorFun().init();
    }

    private JFrame frame;

    public void init() {
        frame = new JFrame("ColorFun");
        frame.setSize(500, 300);
        JButton button1 = new JButton("Red!");
        JButton button2 = new JButton("Blue!");
        button1.addActionListener(this);
        button2.addActionListener(this);
        frame.add(button1, "South");
        frame.add(button2, "South");
        frame.setVisible(true);
    }

    public void actionPerformed(ActionEvent event) {
        if (event.getActionCommand().equals("Red!")) {
            frame.setBackground(Color.BLUE);
        } else {
            frame.setBackground(Color.RED);
}   }   }
```

Summary

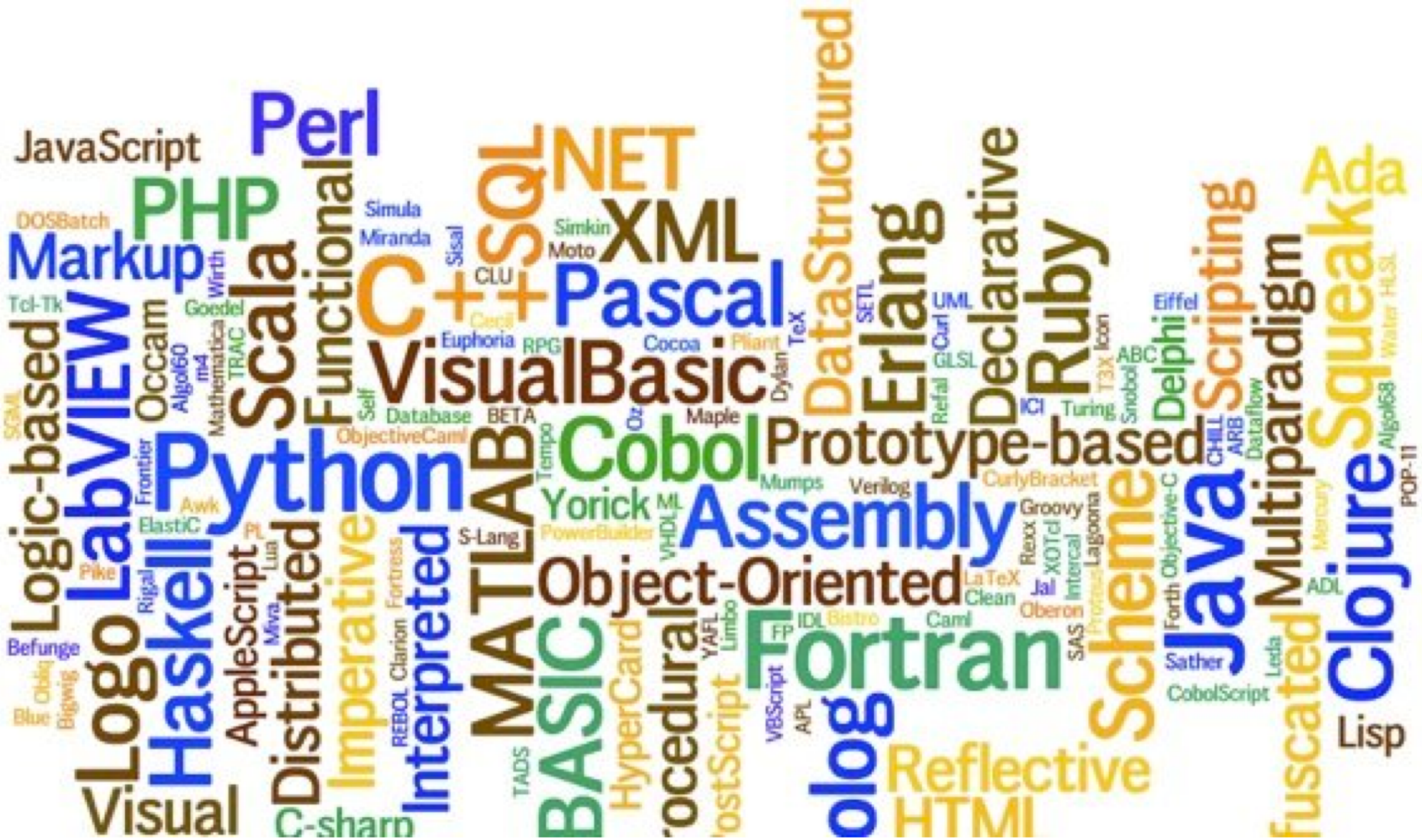
- **Benefits of libraries:**
  - simplify syntax/rough edges of language/API
  - avoid re-writing the same code over and over
  - possible to make advanced programs quickly
  - leverage work of others

- **Drawbacks of libraries:**
  - learn a "dialect" of the language ("ACM Java" vs. "real Java")
  - lack of understanding of how lower levels or real APIs work
  - some libraries can be buggy or lack documentation
  - limitations on usage; e.g., ACM library cannot be re-distributed for commercial purposes

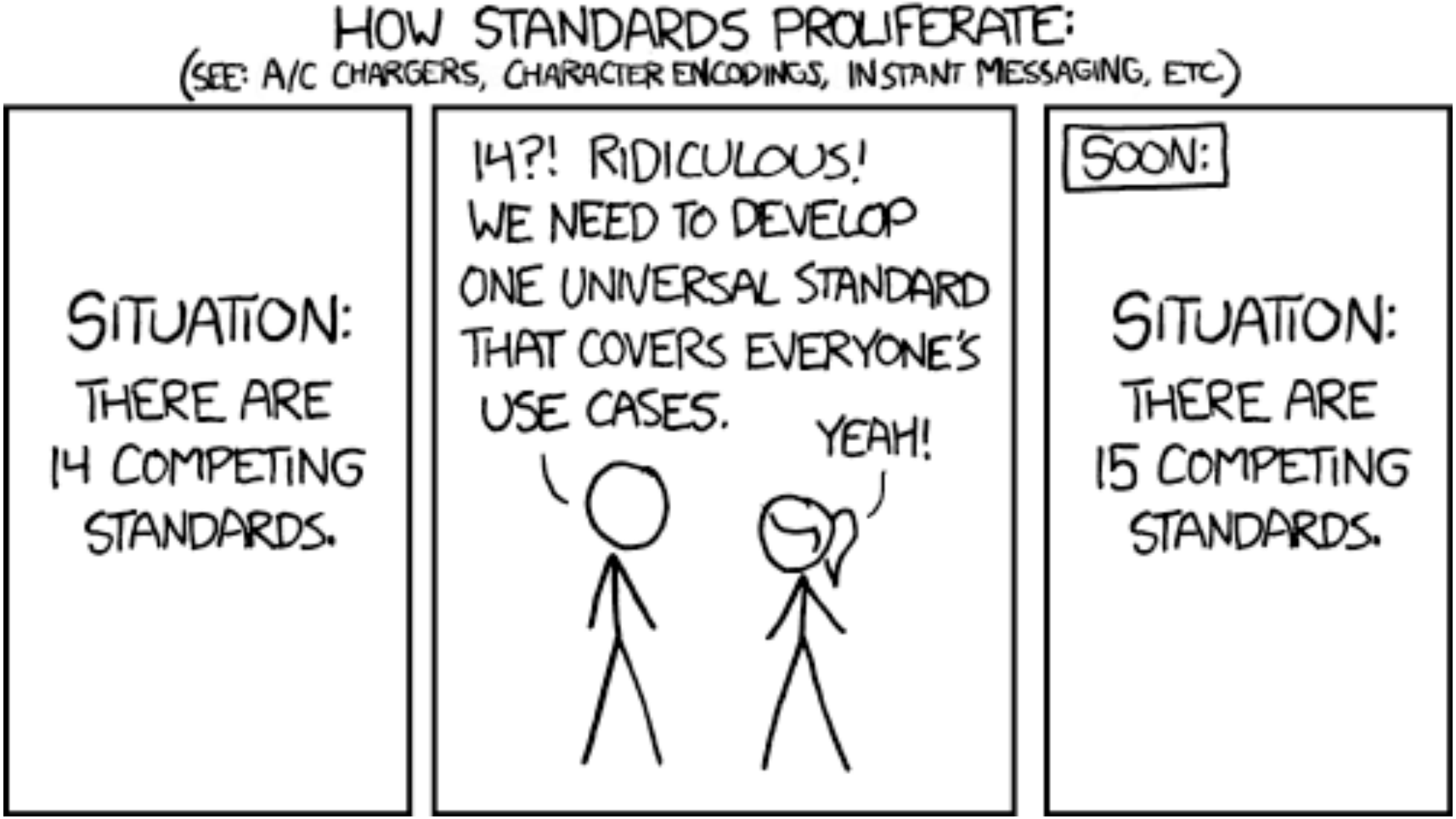
Plan for today

- Life after the ACM Libraries
- Life after Java
- Life after PC programs: Internet applications

Programming Languages



Programming Languages



https://imgs.xkcd.com/comics/standards.png

Java

```
ArrayList<Double> evens = new ArrayList<>();
for (int i = 0; i < 100; i++) {
    if (i % 2 == 0) {
        evens.add(i);
    }
}
println(evens);
```

prints [2, 4, 6, 8, 10, 12, …]

C++

```
Vector<double> evens;
for (int i = 0; i < 100; i++) {
    if (i % 2 == 0) {
        evens.add(i);
    }
}
cout << evens << endl;
```

prints [2, 4, 6, 8, 10, 12, …]

Javascript

```
var evens = []
for (var i = 0; i < 100; i++) {
    if (i % 2 == 0) {
        evens.push(i)
    }
}
console.log(evens)
```

prints [2, 4, 6, 8, 10, 12, …]

Python

```
evens = []
for i in range(100):
    if i % 2 == 0:
        evens.append(i)
print(evens) // this is Python 3 syntax
```

prints [2, 4, 6, 8, 10, 12, …]

Plan for today

- Life after the ACM Libraries
- Life after Java
- Life after PC programs: Internet applications

Programs and the Internet

How does your phone communicate with Facebook?

Programs and the Internet

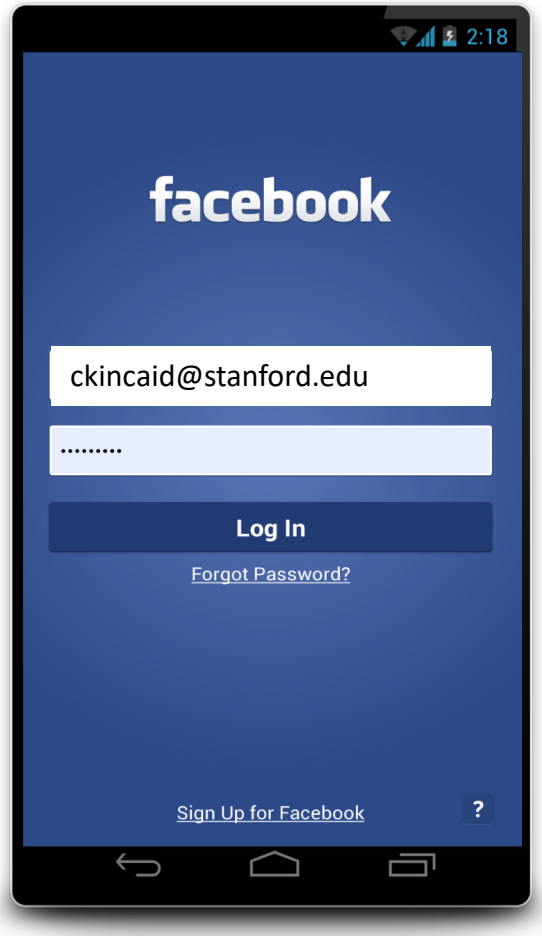
The Java program on your **phone** talks to the Java program at **Facebook**.

Facebook

Facebook Server

* Android phones run Java. So do Facebook servers!

Facebook

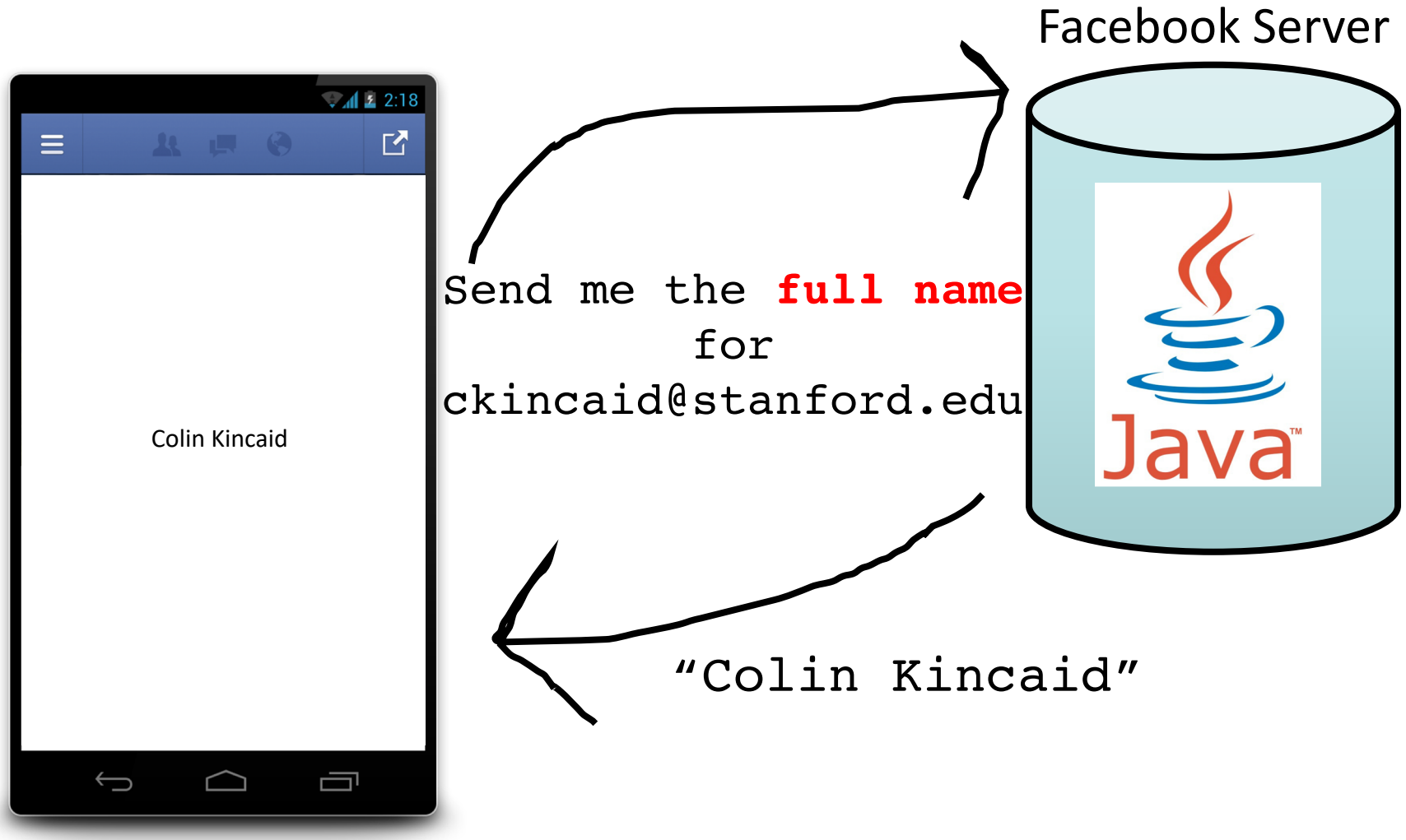


Facebook Server

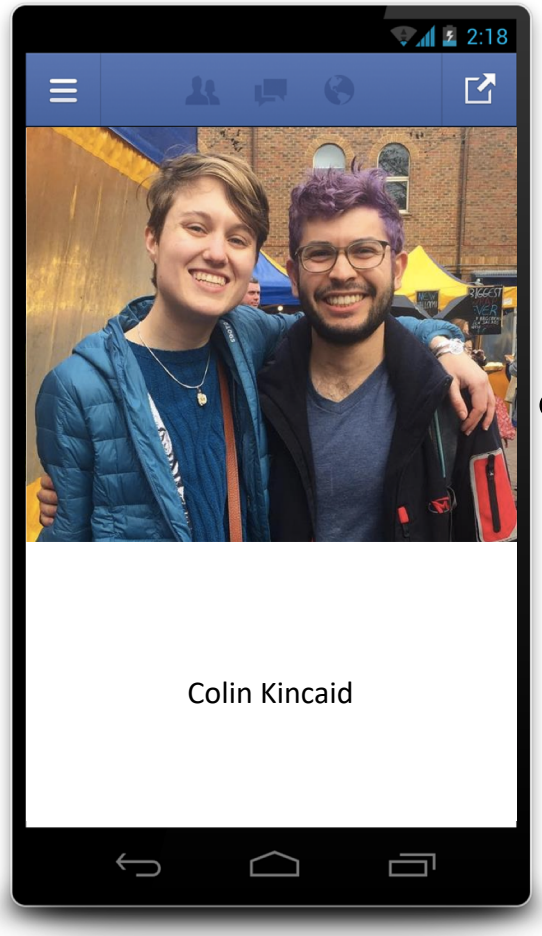
Is this login ok?

Confirmed.
ckincaid@stanford.edu
is now logged in.

Facebook


Facebook Server
2:18
Colin Kincaid
Send me the full name for ckincaid@stanford.edu
"Colin Kincaid"
Java

Facebook



Send me the **cover photo** for ckincaid@stanford.edu

Facebook Server

Programs and the Internet

There are two types of internet programs: **clients** and **servers**.

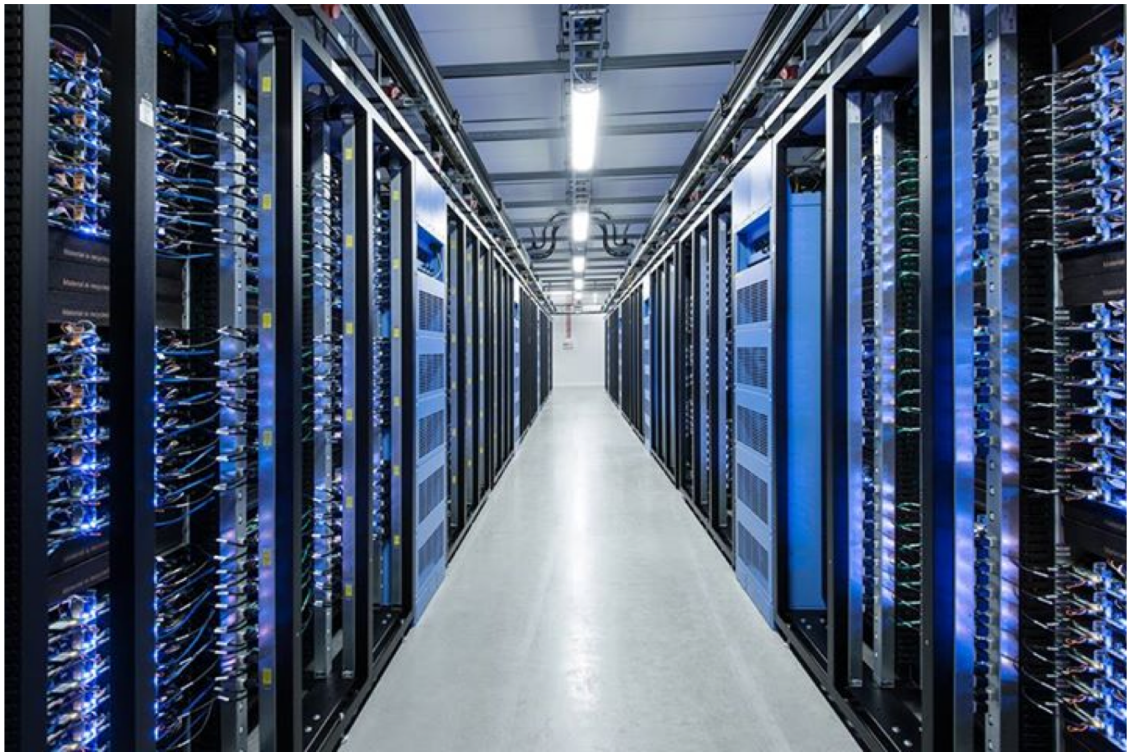
Programs and the Internet

Clients send **requests** to **servers**, who **respond** to those requests.

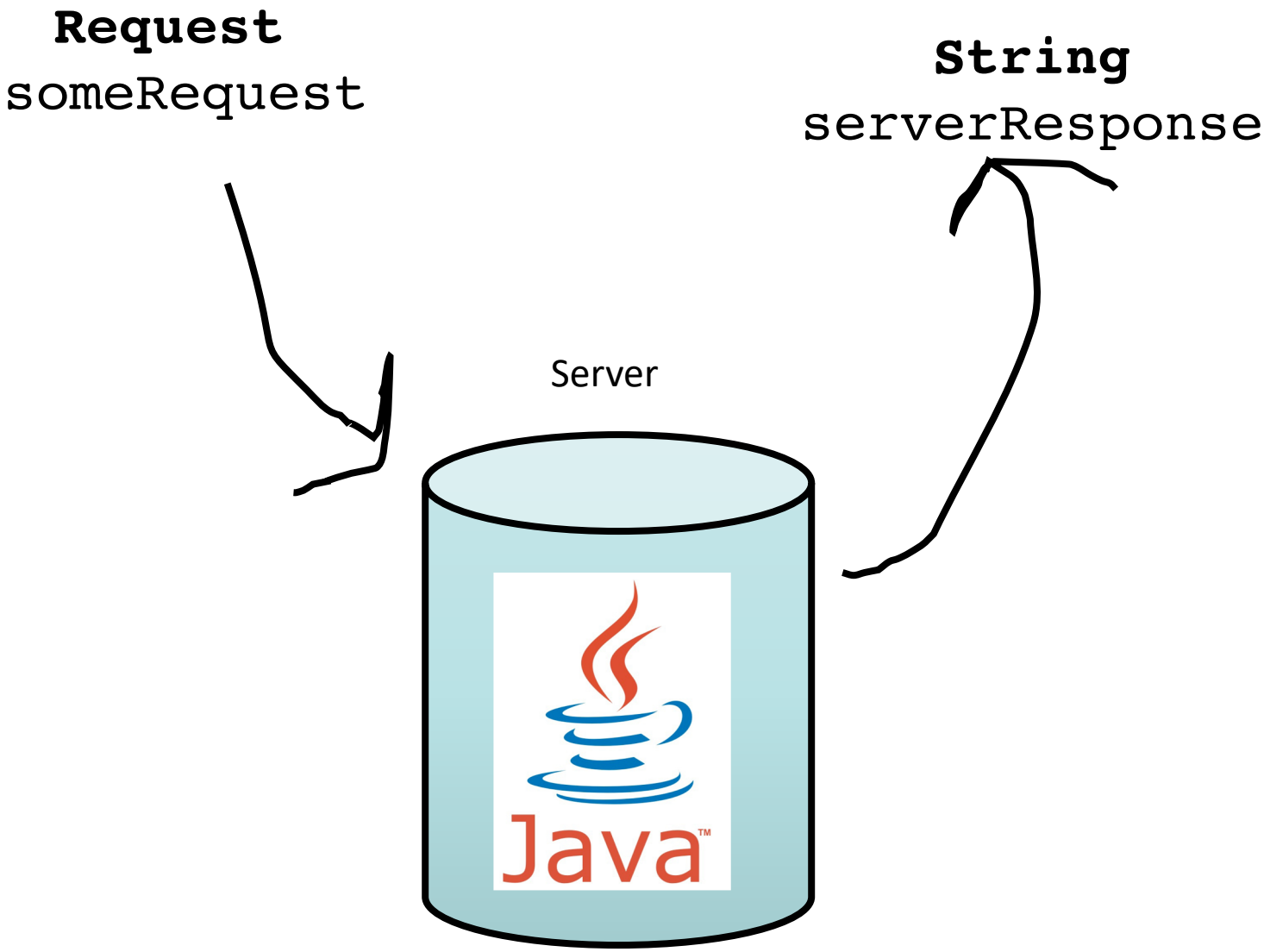
Servers Are Computer Programs!

Facebook Server

=



Servers

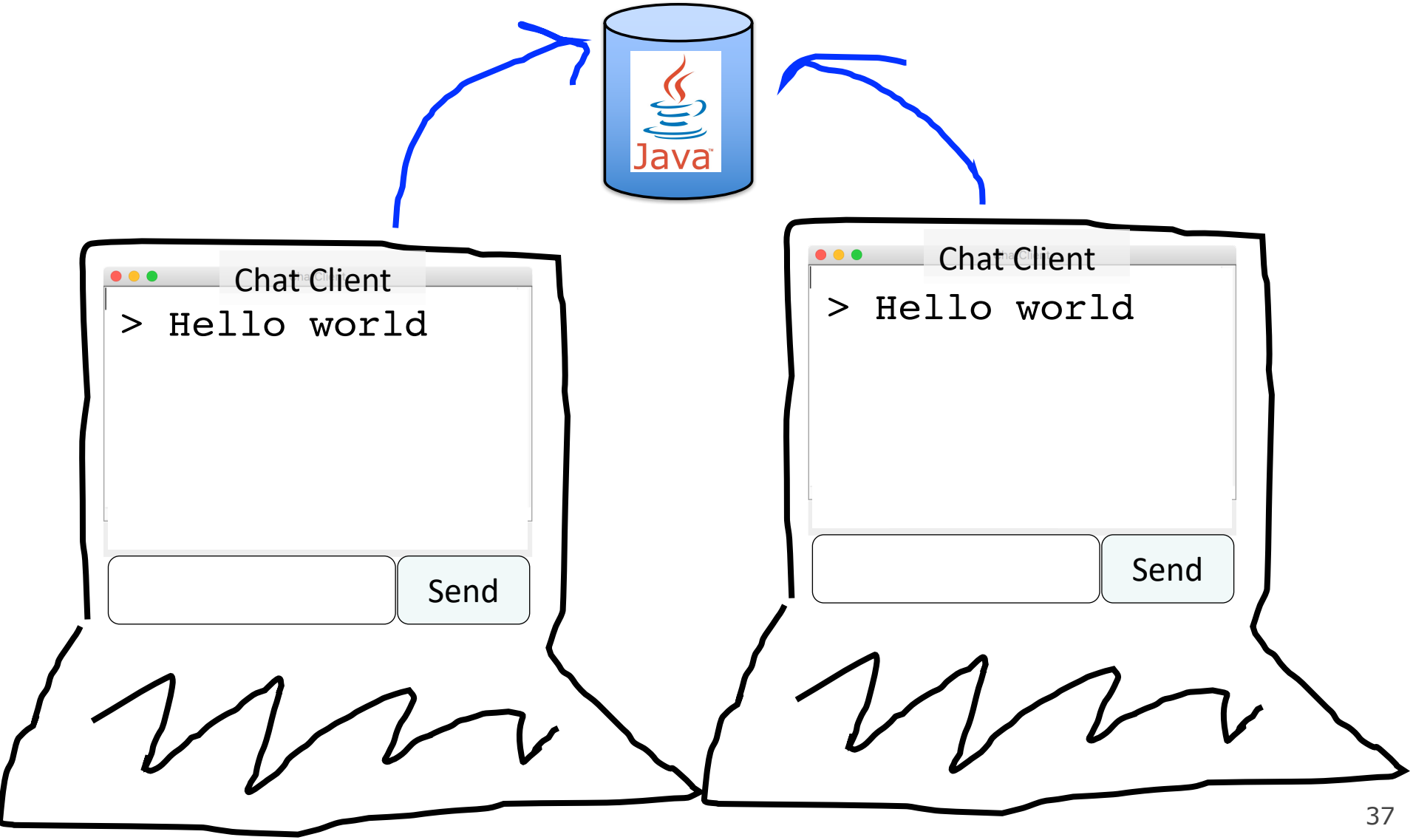


What is a Request?

```
/* Request has a command */
String command;

/* Request has parameters */
HashMap<String,String> params;
```

Clients


Java
Chat Client
> Hello world
Send
Chat Client
> Hello world
Send

Trivia URL

http://ebcb2471.ngrok.io

Recap

- Life after the ACM Libraries
- Life after Java
- Life after PC programs: Internet applications

Tomorrow: Intro to Machine Learning!