CS4/MSc Distributed Systems
Tutorial 1: Hello, Java!

Stephen Gilmore
stg@dcs.ed.ac.uk

Tutorial: M 12, Room 5325
Office hour: Th 2, Room 1608

Yussuf Abu-Shaaban
yna@dcs.ed.ac.uk

Tutorial: M 12, Room 6207
Office hour: M 4, Room 3311

Hello world!

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

- This class is kept in file *Hello.java*
- Compiled with `javac Hello.java` to produce *Hello.class*
- Run with `java Hello` to output `Hello world!`

Invoking methods

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

- This class introduces an *object*, `hello`, of class `String`
- It invokes the method `toUpperCase()` on the `hello` object
- Run with `java Shout` to produce `HELLO, WORLD!`

Common Java programming error (1)

```
class BadHello {
    public static void main (String[] args) {
        String hello = null;
        System.out.println(hello.toUpperCase());
    }
}
```

- This program fails with a “null pointer exception” when invoking the method `toUpperCase()`
- An object must be constructed before we can invoke methods upon it.

Object orientation

```
class OOHello {
    public OOHello() {
        System.out.println("OO, hello world!");
    }
    public static void main(String[] args) {
        OOHello ooh = new OOHello();
    }
}
```

- Classes have *constructors* to make a new object instance
- The constructor has the same name as the class itself and is invoked by **`new`**

Interfaces and inheritance

```
class StructuredHello {
    interface Hello {
        public void hello();
    }
    public static class SuperClass implements Hello {
        public void hello() {
            System.out.println("Hello, world!");
        }
    }
    public static class SubClass extends SuperClass {}
    public static void main(String[] args) {
        SubClass sc = new SubClass();
        sc.hello();
    }
}
```


Command line arguments

```
class SayHello {
    public static void main (String[] args) {
        if (args.length != 1) {
            System.out.println("Usage: java SayHello st");
            System.exit(1);
        }
        System.out.println("Hello, " + args[0] + "!");
    }
}
```

- The main method accepts an array of command-line arguments
- Arrays have a `length` field. Indexing is from zero.
- Run with `java SayHello everyone`

Common Java programming error (2)

```
class BadSayHello {
    public static void main (String[] args) {
        // Forgot to check array access was OK ...
        System.out.println("Hello, " + args[0] + "!");
    }
}
```

- `java BadSayHello` — this command would fail
- An “array index out of bounds” exception is thrown if the array `args` is empty

Type conversions

```
class AddOne {
    public static void main (String[] args) {
        if (args.length != 1) {
            System.out.println("Usage: java AddOne n");
            System.exit(1);
        }
        int i = Integer.parseInt(args[0]);
        System.out.println(i + 1);
    }
}
```

- Invokes the *static method* `parseInt()` of the `Integer` class to convert a `String` to an `int`

Hello server

```
import java.net.*;
class HelloServer {
    public static void main (String[] args)
        throws java.io.IOException {
        int portnum = 5597;
        ServerSocket sock = new ServerSocket(portnum);
        Socket client = sock.accept();
        client.getOutputStream().write('H');
    }
}
```

- Run this on `twain.dai.ed.ac.uk` with `java HelloServer`

Hello client

```
import java.net.*;
class HelloClient {
    public static void main (String[] args)
            throws Exception {
        int portnum = 5597;
        String hostname = "twain.dai.ed.ac.uk";
        Socket sock = new Socket(hostname, portnum);
        System.out.println(hostname + " said " +
            (char)sock.getInputStream().read());
    }
}
```

- Run this on `scar.dcs.ed.ac.uk` with `java HelloClient`

Packages

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

- Kept in: *eduni/distributedsystems/Hello.java*
- To compile: `javac eduni/distributedsystems/Hello.java`
- To run: `java eduni.distributedsystems.Hello`

Importing classes from packages

```
import eduni.distributedsystems.*;
class Driver {
    public static void main (String[] args) {
        Hello.main(null);
    }
}
```

- Imports all the classes from `eduni.distributedsystems`
- Invokes the *static method* `main()` of the `Hello` class
- Arrays are objects in Java and **`null`** is the null object pointer

The class path

- A *path* is a list of directories to be searched in order.
- A *class path* is a list of directories containing classes.
- If `/home/cs4/www/disys` contains the `eduni` folder, use

```
export CLASSPATH=.:/home/cs4/www/disys
javac Driver.java
java Driver
```

or

```
javac -classpath .:/home/cs4/www/disys Driver.java
java -classpath .:/home/cs4/www/disys Driver
```


Java resources

- Java 2 SDK, Standard Edition Documentation, `http://java.sun.com/j2se/1.3/docs/index.html`
- The Java tutorial, `http://java.sun.com/docs/books/tutorial/index.html`
- The Java 2 Enterprise Edition tutorial, `http://java.sun.com/j2ee/tutorial/`
- Java blueprints, `http://java.sun.com/blueprints/`