A first taste of Java

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming Concepts using Java

Week 2

Getting started

The C Programming Language, Brian W Kernighan, Dennis M Ritchie

The only way to learn a new programming language is by writing programs in it. The first program is the same for all languages.

Print the words

```
hello, world
```

This is a big hurdle; to leap over it you have to create the program text somewhere, compile it successfully, load it, run it, and find out where your output went. With these mechanical details mastered, everything else is comparatively easy

Getting started

The C Programming Language, Brian W Kernighan, Dennis M Ritchie

The only way to learn a new programming language is by writing programs in it. The first program is the same for all languages.

Print the words

```
hello, world
```

This is a big hurdle; to leap over it you have to create the program text somewhere, compile it successfully, load it, run it, and find out where your output went. With these mechanical details mastered, everything else is comparatively easy

- In Python

```
print("hello, world")
```

Getting started

The C Programming Language, Brian W Kernighan, Dennis M Ritchie

The only way to learn a new programming language is by writing programs in it. The first program is the same for all languages.

Print the words

`hello, world`

This is a big hurdle; to leap over it you have to create the program text somewhere, compile it successfully, load it, run it, and find out where your output went. With these mechanical details mastered, everything else is comparatively easy

- In Python

```
print("hello, world")
```

- . . . C

```
#include <stdio.h>
main()
{
  printf("hello, world\n");
}
```

Getting started

The C Programming Language, Brian W Kernighan, Dennis M Ritchie

The only way to learn a new programming language is by writing programs in it. The first program is the same for all languages.

Print the words

`hello, world`

This is a big hurdle; to leap over it you have to create the program text somewhere, compile it successfully, load it, run it, and find out where your output went. With these mechanical details mastered, everything else is comparatively easy

- In Python

```
print("hello, world")
```

- . . . C

```
#include <stdio.h>
main()
{
  printf("hello, world\n");
}
```

- . . . and Java

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

Why so complicated?

- Let's unpack the syntax

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

Why so complicated?

- Let's unpack the syntax
- All code in Java lives within a class
  - No free floating functions, unlike Python and other languages
  - Modifier `public` specifies visibility

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

Why so complicated?

- Let's unpack the syntax
- All code in Java lives within a class
  - No free floating functions, unlike Python and other languages
  - Modifier `public` specifies visibility
- How does the program start?
  - Fix a function name that will be called by default
  - From C, the convention is to call this function `main()`

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

Why so complicated . . .

- Need to specify input and output types for `main()`
  - The signature of `main()`
  - Input parameter is an array of strings; command line arguments
  - No output, so return type is `void`

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

Why so complicated . . .

- Need to specify input and output types for `main()`
  - The signature of `main()`
  - Input parameter is an array of strings; command line arguments
  - No output, so return type is `void`
- Visibility
  - Function has be available to run from outside the class
  - Modifier `public`

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

Why so complicated . . .

- Availability
  - Functions defined inside classes are attached to objects
  - How can we create an object before starting?
  - Modifier `static` — function that exists independent of dynamic creation of objects

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

Why so complicated . . .

- The actual operation
  - System is a public class

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

Why so complicated . . .

- The actual operation
  - `System` is a public class
  - `out` is a stream object defined in `System`
    - Like a file handle
    - Note that `out` must also be `static`

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

Why so complicated . . .

- The actual operation
  - System is a public class
  - out is a stream object defined in System
    - Like a file handle
    - Note that out must also be static
  - println() is a method associated with streams
    - Prints argument with a newline, like Python print()

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

Why so complicated . . .

- The actual operation
  - System is a public class
  - out is a stream object defined in System
    - Like a file handle
    - Note that out must also be static
  - println() is a method associated with streams
    - Prints argument with a newline, like Python print()
- Punctuation {, }, ; to delimit blocks, statements
  - Unlike layout and indentation in Python

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

Compiling and running Java code

- A Java program is a collection of classes

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

Compiling and running Java code

- A Java program is a collection of classes
- Each class is defined in a separate file with the same name, with extension `java`
  - Class `helloworld` in `helloworld.java`

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

Compiling and running Java code

- A Java program is a collection of classes
- Each class is defined in a separate file with the same name, with extension `java`
  - Class `helloworld` in `helloworld.java`

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

- Java programs are usually interpreted on Java Virtual Machine (JVM)
  - JVM provides a uniform execution environment across operating systems
  - Semantics of Java is defined in terms of JVM, OS-independent
  - "Write once, run anywhere"

Compiling and running Java code

- javac compiles into JVM bytecode
    - javac helloworld.java creates bytecode file helloworld.class

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

Compiling and running Java code

- javac compiles into JVM bytecode
  - javac helloworld.java creates bytecode file helloworld.class
- java helloworld interprets and runs bytecode in helloworld.class

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

Compiling and running Java code

- javac compiles into JVM bytecode
  - javac helloworld.java creates bytecode file helloworld.class
- java helloworld interprets and runs bytecode in helloworld.class
- Note:
  - javac requires file extension .java
  - java should not be provided file extension .class
  - javac automatically follows dependencies and compiles all classes required
    - Sufficient to trigger compilation for class containing main()

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

Summary

- The syntax of Java is comparatively heavy
- Many modifiers: unavoidable overhead of object-oriented design
  - Visibility: `public` vs `private`
  - Availability: all functions live inside objects, need to allow `static` definitions
  - Will see more modifiers as we go along
- Functions and variable types have to be declared in advance
- Java compiles into code for a virtual machine
  - JVM ensures uniform semantics across operating systems
  - Code is guaranteed to be portable