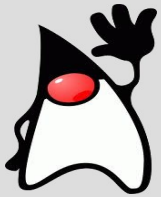
JAVA

http://www.julien-roche.fr



- A quick tour of java

- Implementation in Java

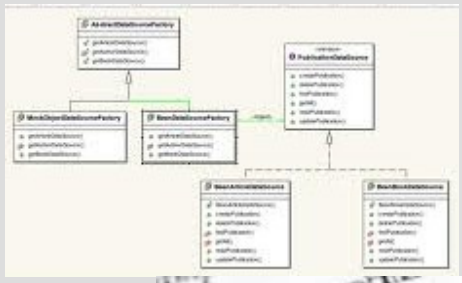
- make a presentation (4 or 5 students)

- java project (4 or 5 students)

A quick tour of java

- Introduction
- Primitive Types
- Object Definition
- Object Instantiation
- Object Access and Message Passing
- Representational independence
- Overloading
- Initialization and Constructors
- Expressions, Statements, and Control-flow Mechanisms
- Arrays

Implementation in Java



- **Make a Timetable application for Mobile phone working with Android.**

- **And more**

make a presentation

A team of 4 or 5 students will choose one of the following subjects and make a short presentation about it (10 minutes). Each subject will be illustrated with java code.

- **Inheritance**
- **Polymorphism**
- **Exception Handling**
- **Input and Output Operations**
- **Graphical Interfaces and Windows**
- **Applets and Loaders**
- **Java Servlets**
- **Object Serialization and Remote Method Invocation**
- **Java Database Connectivity**

Java project

- **A team of 4 or 5 students will develop a software in Java.**
- **The team will provide the documentation in English related to it.**
- **Students can choose the subject but the program must include at least an XML parser.**

So it can be a tool, a game...

2013 IGF Student Showcase Competition

Who Is Eligible:	Open to all student game developers worldwide, including student mod makers (see student-specific rules).
Competing For:	8 'Student Showcase' winners for top game ($1000 cash prize), and for the second year, an overall 'Best Student Game' ($3,000 cash prize).
Entry Fee:	None.
Submissions Due:	Submissions are now closed.

The IGF's Student Showcase, for which the entry deadline was **October 31st, 2012** at 11.59pm PDT, will highlight a total of eight games this year, each of which will receive:

IGF Student Showcase Winner ($1000)

In addition, there will be a prize for best overall IGF Student Game awarded as part of the IGF Awards, with the finalists comprising all of the Student Showcase winners:

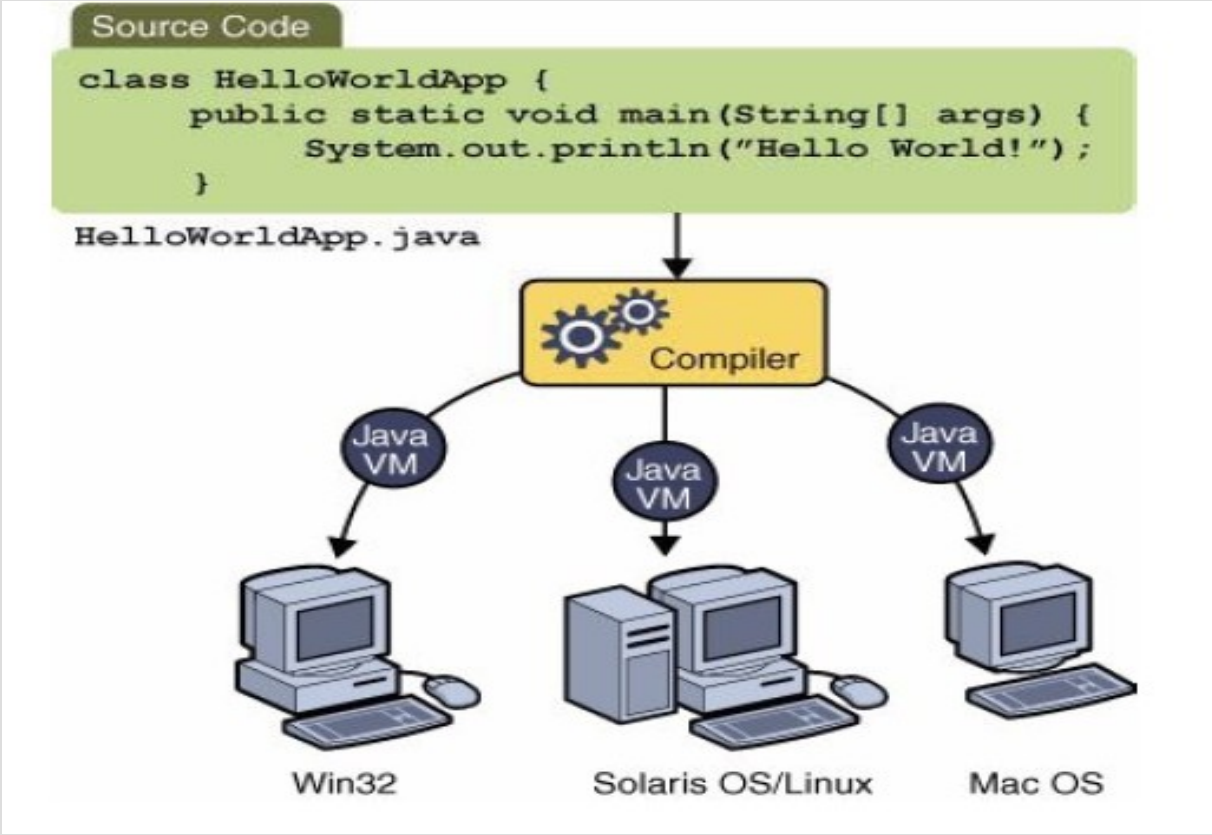
Best Student Game ($3,000)

[Please note that both games using middleware engines and mods of existing games are eligible for the Student Showcase, but the judges will take whether the engine was coded from scratch into account when judging.]

Introduction

Java is

- **Easy to learn**
- **Portable**

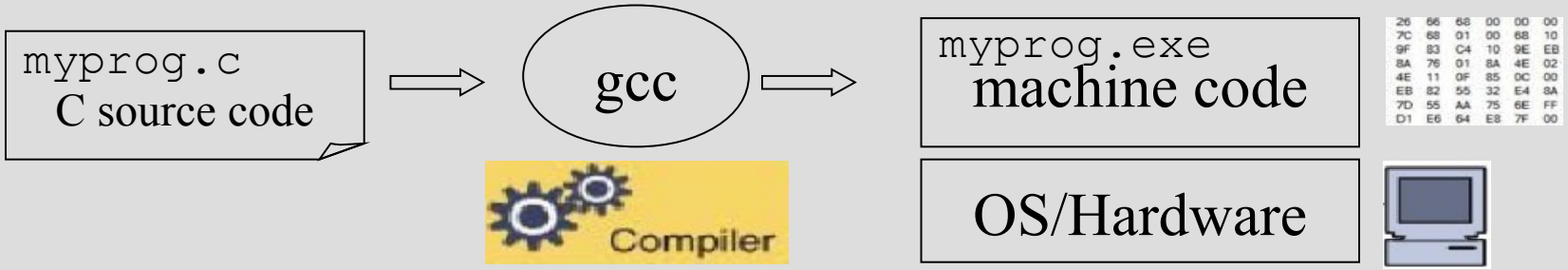


Introduction

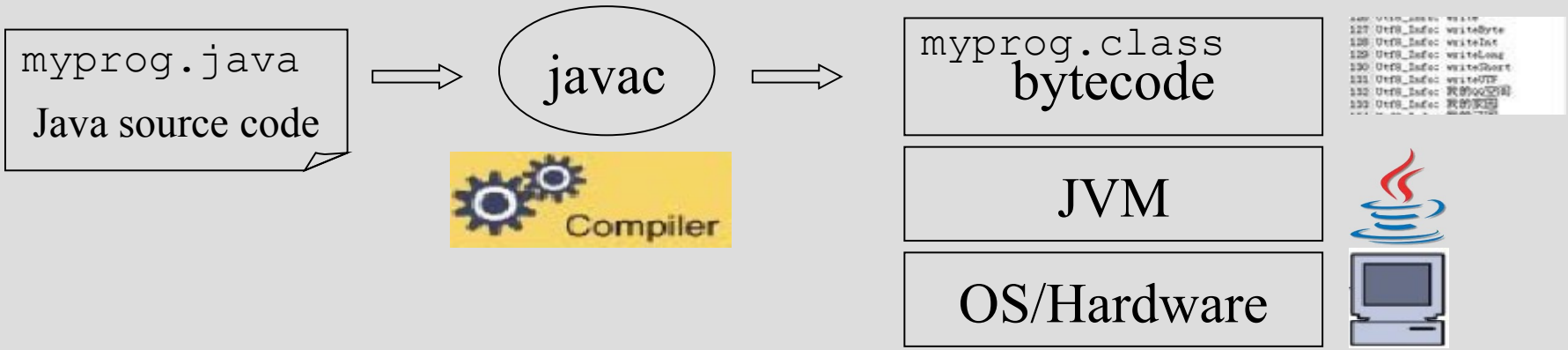
- JVM stands for Java Virtual Machine
- Unlike other languages, Java “executables” are executed on a CPU (Central Processing Unit) that does not exist.

Introduction

Platform Dependent



Platform Independent



Introduction

1) Installation of Java

Java might already be installed on your machine. You can test this by opening a console (if you are using Windows: Win+R, enter *cmd* and press Enter) and by typing in the following command:

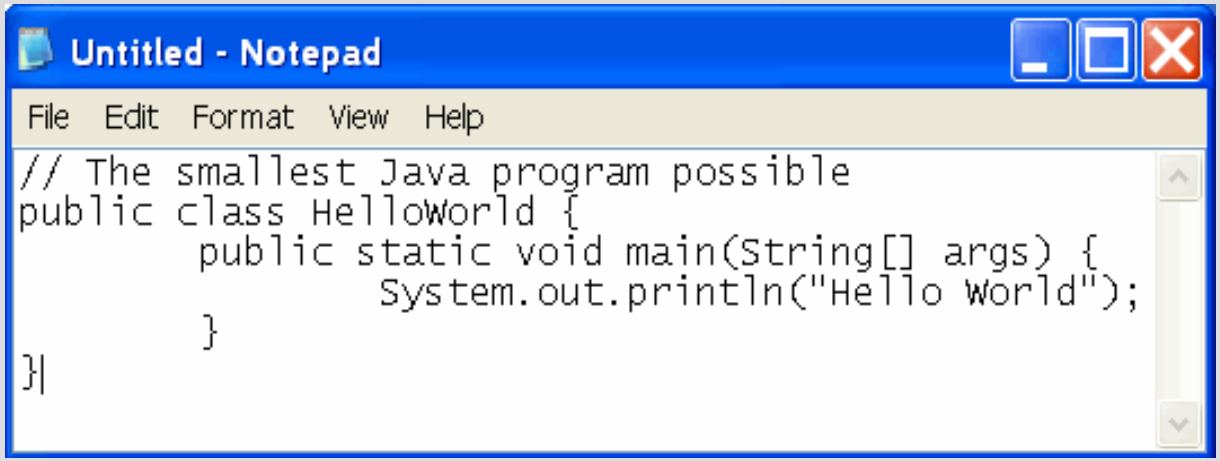
```
java -version
```

If Java is correctly installed, you should see some information about your Java installation. If the command line returns the information that the program could not be found, you have to install Java. The central website for installing Java is the following URL:

```
http://java.com
```

Introduction

2) Your first Java program



Save the source code in your *javadir* directory with the `HelloWorld.java` filename. The name of a Java source file must always equals the class name (within the source code) and end with the `.java` extension. In this example the filename must be `HelloWorld.java` because the class is called `HelloWorld`.

Introduction

3) Compile your Java program

Switch to the command line, e.g. under Windows Start-> Run -> cmd. Switch to the *javadir* directory with the command cd javadir, for example in my case cd c:\temp\java. Use the command dir to see that the source file is in the directory.

```
javac HelloWorld.java
```

The directory contains now a file "HelloWorld.class". If you see this file you have successfully compiled your first Java source code into bytecode.

Introduction

4) Run your Java program

Run -> cmd. Switch to the directory jardir.

To run your program type in the command line:

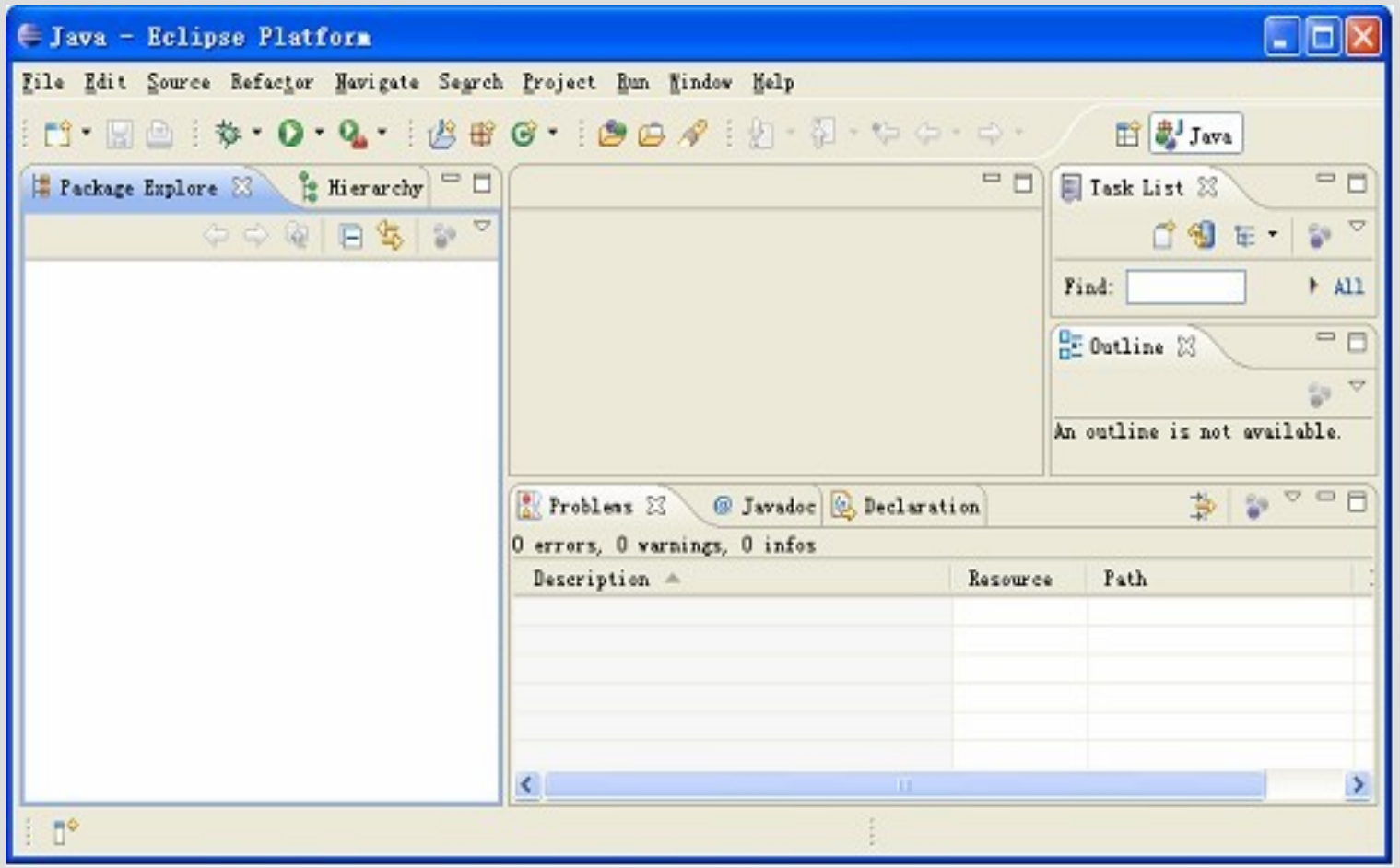
```
java HelloWorld
```

```
C:\temp\java>java HelloWorld
Hello World

C:\temp\java>_
```

Introduction

5) Integrated Development Environment (IDE)



Primitive Types

Primitive Type	Size	Minimum Value	Maximum Value
char	16-bit	Unicode 0	Unicode $2^{16}-1$
byte	8-bit	-128	+127
short	16-bit	-2^{15} *(-32,768)*	$+2^{15}-1$ *(32,767)*
int	32-bit	-2^{31} *(-2,147,483,648)*	$+2^{31}-1$ *(2,147,483,647)*
long	64-bit	-2^{63} *(-9,223,372,036,854,775,808)*	$+2^{63}-1$ *(9,223,372,036,854,775,807)*
float	32-bit	32-bit IEEE 754 floating-point numbers	
double	64-bit	64-bit IEEE 754 floating-point numbers	
boolean	1-bit	`true` or `false`	

Note:
Primitive type always begin with lower-case

Object Definition

```
public class Counter {
    // Attribute and method declarations
}
```

Object Definition

```
public class Counter {
      int number;

      void add(){
            number = number +1;
      }
}
```

- Object attributes are component objects or primitive types used to represent the object.

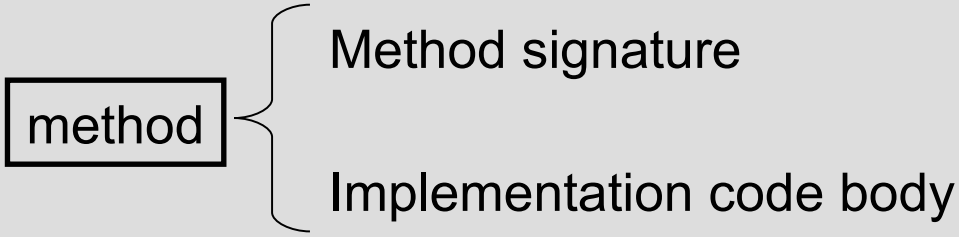
- An instance method manipulates the object by altering its attribute values.

Variable Definitions

```
Counter myCounter;
```

- "Counter" is the type name
- "myCounter" is the variable name

Methods




```
T m(R1 p1, R2 p2, ... Rn pn){
   //code body
}
```

m *is the method name,* ***T*** *is the return type, with* ***Rn*** *et* ***pn*** *being parameter types and names, respectively.*

Object Instantiation

```
Counter carPark; // Variable definition
Counter entrance, exitDoor; // Variable definition

carPark= new Counter(); // Object instantiation
entrance= new Counter(); // Object instantiation
exitDoor= new Counter(); // Object instantiation
```

► ***The expression « new Counter() » returns a newly created instance of Counter class.***

Object Access and Message Passing

```
entrance.add();
exitDoor.add();
int totalDoors= entrance.number+exitDoor.number;
```

► ***Attributes and methods of an object are accessed via the qualification operator « . »***

Representational Independance

```
public class Counter {
    private int number=0;

    void add(){
        number = number +1;
    }

    public int getNumber() {
        return number;
    }

    public void setNumber(int number) {
        this.number = number;
    }

}
```

► ***Attributes and methods of an object are accessed via the qualification operator « . »***

Overloading

```
void add(){
    number = number +1;
}

public void add(int x){
    number=number+1;
}
```

Methods share the same name as long as they may distinguished either by:

- **The number of parameters, or**
- **Different parameter types.**

Initialization and Constructors

```
public class Counter {
    private int number;

    public Counter(){
        number=0;
    }

}
```

Expressions, Statements, and Control-flow Mechanisms

a) Arithmetic Operators

The arithmetic operators in Java include the common addition «+», substraction « - », multiplication « * » and division « / » operations.

The « % » operator returns the remainder of integer division.

```
int a=13;
int b=3;

// a / b returns result 4
// a % b returns result 1
```

Expressions, Statements, and Control-flow Mechanisms

b) Logical Operators

- **&&(and)**
- **||(or)**
- **!(not)**

c) Relational Operators

==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Expressions, Statements, and Control-flow Mechanisms

d) Assignment

```
int a,b;

a = b = 2;
```

```
int f,g;
f = 6;
g = f++;
// g has 6
// f has 7
```

```
int f,g;
f = 6;
g = ++f;
// g has 7
// f has 7
```

```
int f,g;
f = 6;
g = f--;
// g has 6
// f has 5
```

```
int f,g;
f = 6;
g = --f;
// g has 5
// f has 5
```

Expressions, Statements, and Control-flow Mechanisms

e) Typecast

```
Object x = new Counter();

Counter c;

 c = (Counter) x;
```

Expressions, Statements, and Control-flow Mechanisms

f) Control-flow statement

- **Conditional Statements**

```
if (E) S;

if (E) S; else E;

switch (E) {
        c1: S1; break;
        c2: S2; break;
        c3: S3; break;
        default: Sd;
}
```

Expressions, Statements, and Control-flow Mechanisms

f) Control-flow statement

- **Iterative Statements**

while (E) S;

---

do {

R;

} ***while*** (F);

For (Q;T;S) R**;**

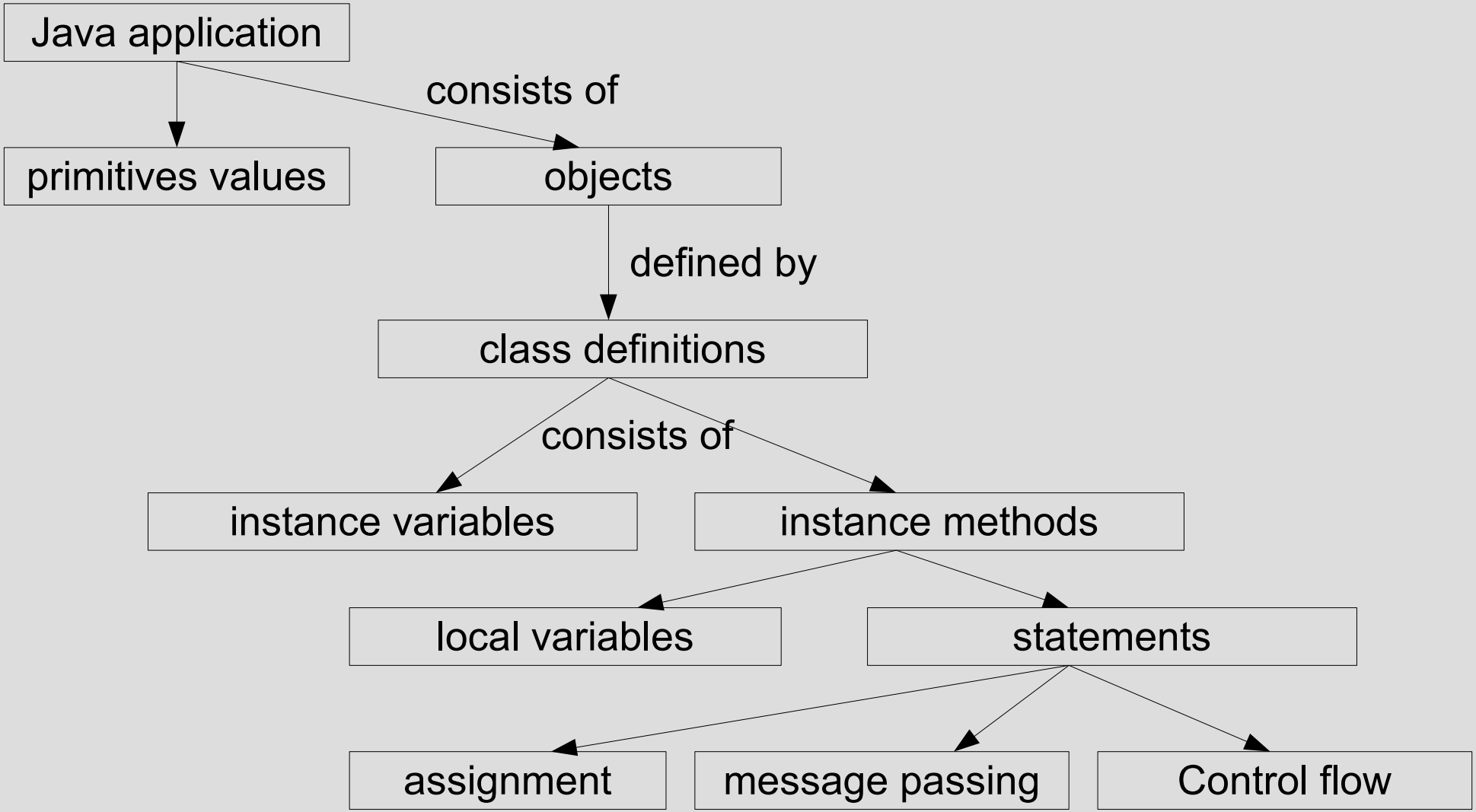
for (int i=0;i<5;i++) System.out.print(i); //01234

Arrays

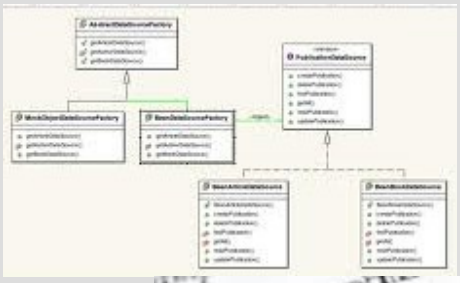
```
Counter gates[];
gates = new Counter[8];

for (int i=0;i<8;i++){
    gates[i]=new Counter();
}
```

Summary


Java application
consists of
primitives values
objects
defined by
class definitions
consists of
instance variables
instance methods
local variables
statements
assignment
message passing
Control flow

Implementation in Java



- **Make a Timetable application for Mobile phone working with Android.**