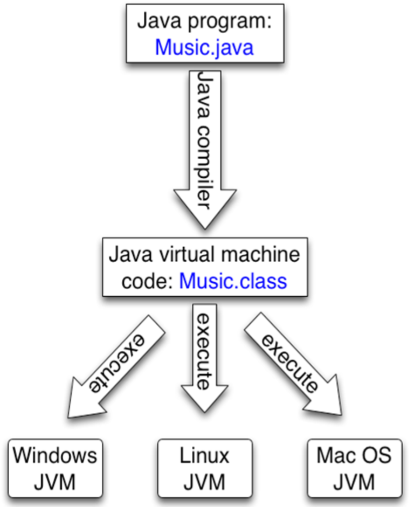
JAVA Express

– Introduction to Java and object-oriented programming in a single lecture –

Guillaume Lopez
Living Environment Laboratory
The University of Tokyo, School of Engineering

C/C++ vs Java


Java program: Music.java

Java compiler

Java virtual machine code: Music.class

execute

execute

execute

Windows JVM

Linux JVM

Mac OS JVM

- C/C++
  - C is a functional language
    - Well at least it is used that way
  - C++ is a hybrid of functional and object oriented
  - Knowing either one helps
- Java
  - A fully object-oriented language (except int, char, etc.)
  - Everything is a **Object** or an instance of one
    - “MobilePhone”, “Keyboard”, “Screen”, “Microphone”, etc.
  - Portability
    - Java compiler produces intermediate code
    - **Java Virtual Machines**
      - Specific for each platform, i.e., Windows, Unix, Android, etc.
    - CORE: Compile Once Run Everywhere
  - Hey, no pointers ☺
    - Even better, you don’t have to de-allocate!

Not the best every time

- Compilation for compactness not performance
  - C/C++ code will outperform Java code
  - JVM can be equipped with Just-In-Time compilers
    - Getting closer, but there yet
    - Look into Java SE HotSpot for more on performance

- Improving performance
  - Write code in a smart way
  - Import what you need and not what you don’t need !
  - Set objects to null when not needed
    - Helps JVM’s internal garbage collection
  - Don’t use synchronized objects if not needed, etc.

Good Match for Java

- Distributed applications
  - Portability and security
  - Code can easily migrate
- Multithreaded environments
  - Good support for concurrent instruction flow
    - Synchronization primitives
  - Data structures support thread queues
- Accessibility aware applications
  - Standalone applications
  - Browser based applets
  - Support for GUI

Object Oriented Programming

- What is a Class?
  - It is a description or a definition
- What is an Object?
  - Instantiation of a class
  - Objects have
    - Name: “MobilePhone”
    - “**Attributes**” that describe the properties, state of the object (local variables)
      - “MobilePhone” object has attributes like “screen” or “keyboard” (that are also objects!)
    - “**Methods**” that describe the object’s dynamics (operations that change the state)
      - “MobilePhone” object has a method to “MakeAPhoneCall”
      - You may also hear about getters and setters
  - Objects related through inheritance
    - An object “Smartphone” inherit from the object “MobilePhone”
    - So at least it should be able to make phone calls + something else !

Simple Java Program

```
// Hello.java
public class Hello {
   public static void main (String[] args) {
        String localVar = new String(“Hello”);
          System.out.println(localVar + “ “);
       // print out first program argument (aka parameter)
       if (args.length >= 1)
             System.out.print(args[0] + " ");
       // if there are more print more
       for (int i=1; i < args.length; i++)
             System.out.print(args[i] + " ");
       System.out.println();
   }
}
```

File name and Class name must be the same

This is one way to print things to the screen.

Control statements to get starting parameters (you should be familiar to it).

Variable declaration. Note String is a class

Each program has a MAIN (just like in C/C++). Note, we force it to be the only one! Note, ***static*** – means there is only one.

What Do You Need?

- A Java Compiler (JDK) and Run Time Entrainment (JRE)
  - See the Android introduction tutorial for link
- Eclipse is a Java development environment (IDE)
  - Useful but not necessary
    - You must learn how to use it on your own
    - Worth the work
    - Great debugging support
- Visit the Java API link (see class webpage)
  - Take a look at the System class
  - Take a look at the String and Integer class
  - The more you look the more you learn!

Compiling & Running at the Prompt

```
> javac Hello.java
```

If no errors, then will produce a file Hello.class

```
> java Hello
```

Should execute the program, now try this:

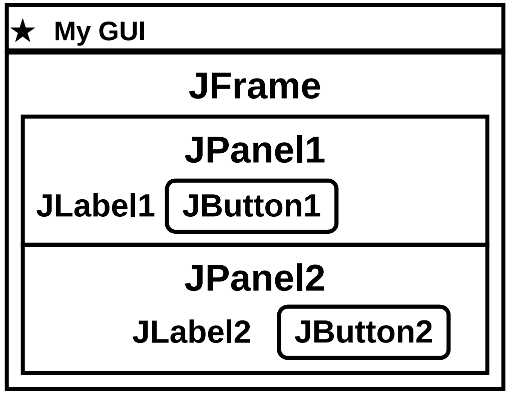
```
> java Hello JAPAN
```

Basic Data Types

- boolean – true or false
- byte – 8-bit integer
- char – 16-bit unsigned integer
- int – 32-bit integer
  - short – 16-bit integer
  - long – 64-bit integer
  - float – 32-bit IEEE 755 single precision number
  - double – 64-bit IEEE 754 double precision number

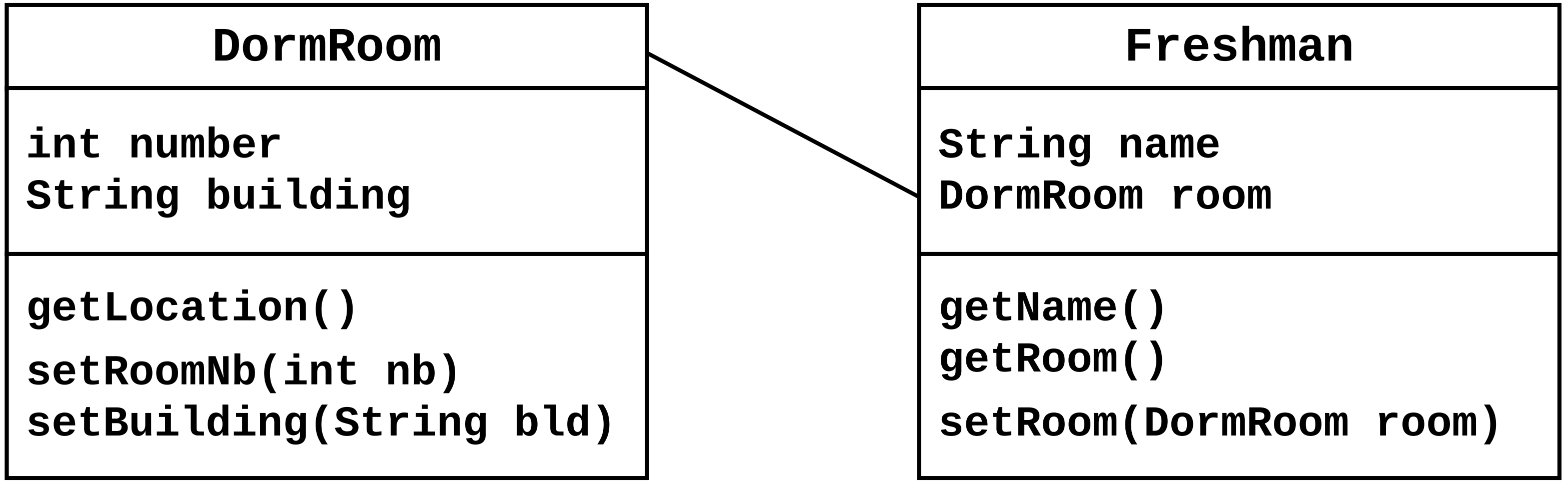
Java Classes Representing Data Types

- Derivatives of the basic ones:
  - Boolean, Long, Character, Short, Integer, Byte, Float, Double, String
  - These have methods that enable easy **manipulation** and **translation**
- More advanced:
  - ArrayList, Vector, Hashtable, …
- Yet more advanced:
  - BufferedReader, InputStreamReader, Socket, …
- Turning Graphical: Swing library
  - JFrame, JPanel, JButton, JLabel,…



Example of Classes’ Relationships

- An object of type A has an instance variable which is an object whose type is B. (A “has a” B.)
- E.g: A Freshman object whose room is of reference type DormRoom
- *UML diagrams* that show instance variables and methods:



DormRoom

```
int number
String building
```

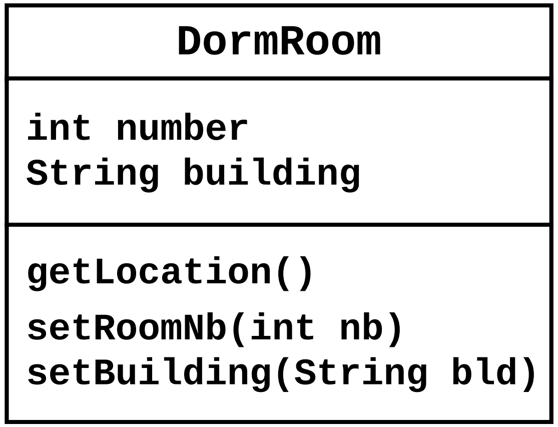
```
getLocation()
setRoomNb(int nb)
setBuilding(String bld)
```

Freshman

```
String name
DormRoom room
```

```
getName()
getRoom()
setRoom(DormRoom room)
```

DormRoom Code and UML



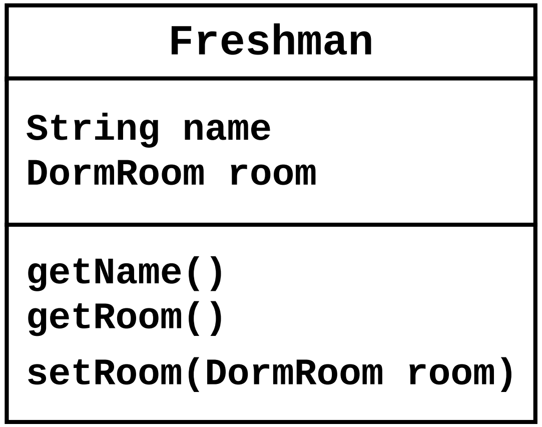
Code

```
public class DormRoom{
  private int num;
  private String bldgName;

  public DormRoom(int n, String b){
    num = n;
    bldgName = b;
  }

  public String getLocation(){
    return num + " " + bldgName;
  }
}
```

Execution

```
> DormRoom room = new DormRoom(208, "Hill");
> room.getLocation()
"208 Hill"
```

Freshman Code and UML



Freshman

String name
DormRoom room

getName()
getRoom()
setRoom(DormRoom room)

Code

```
public class Freshman{
  private String name;
  private DormRoom room;

  public Freshman(String n, DormRoom r){
    name = n;
    room = r;
  }

  public String getName(){ return name;}
  public DormRoom getRoom(){ return room;}
}
```

Execution

```
> DormRoom room = new DormRoom(208, "Hill");
> Freshman f = new Freshman("jo", room);
> f.getName()
"jo"
> f.getRoom().getLocation()
"208 Hill"
```

Things to Know

- Comments
  - // bla bla bla
    - or
  - /* bla
    bla
    ….
    */
- Documentation
  - /**
    Long explanation goes here….
    */

Code Esthetics and Conventions

- Classes should begin with upper letter case
- Variables should begin with lower letter case
  - Class variables should begin with an underscore
- Use indentation
- Use comments
- Be consistent, for example
  - Use i for top layer for-loop index
  - Use j for 2nd layer for-loop index
  - Use k for 3rd layer for-loop index
- If you need to use a temporary variable, put “temp” in its name …

The End

Now you know!

Sources:

- "Java – a Crash Course in a Single Lecture" by Peter M. Musial Univ of Puerto Rico (http://ccom.uprrp.edu/~pmusial/)
- "Introduction to programming with Java, for beginners" by Diana Palsetia, University of Pennsylvania (http://www.seas.upenn.edu/~pfpcse/)
- http://www.acm.org/crossroads/xrds4-2/ovp42.html
- http://java.sun.com/javase/technologies/hotspot