Advanced Programming
CSE 201

Instructor: Sambuddho

(Semester: Monsoon 2024)

Week 1 - Introduction and Basics

Introduction

- What is this course about ? - Introduction to OOP (using Java)
- Course outline:
  - Basics of Java.
  - Classes and objects.
  - Inheritance.
  - Polymorphism.
  - Abstract classes.
  - Interfaces.
  - Exceptions and exception handling.
  - Collections.
  - I/O operations.
  - Multithread programming and synchronizations
  - Basics of design patterns.

Introduction

- Evaluations:
  - Assignments: 20% (total 4 assignments)
  - Quizzes: 10% (best 5 out of 6; 3 before midsem and 3 after midsem)
  - Midsem exam: 25%
  - Final exam: 25%
  - Project: 20% (in pairs)

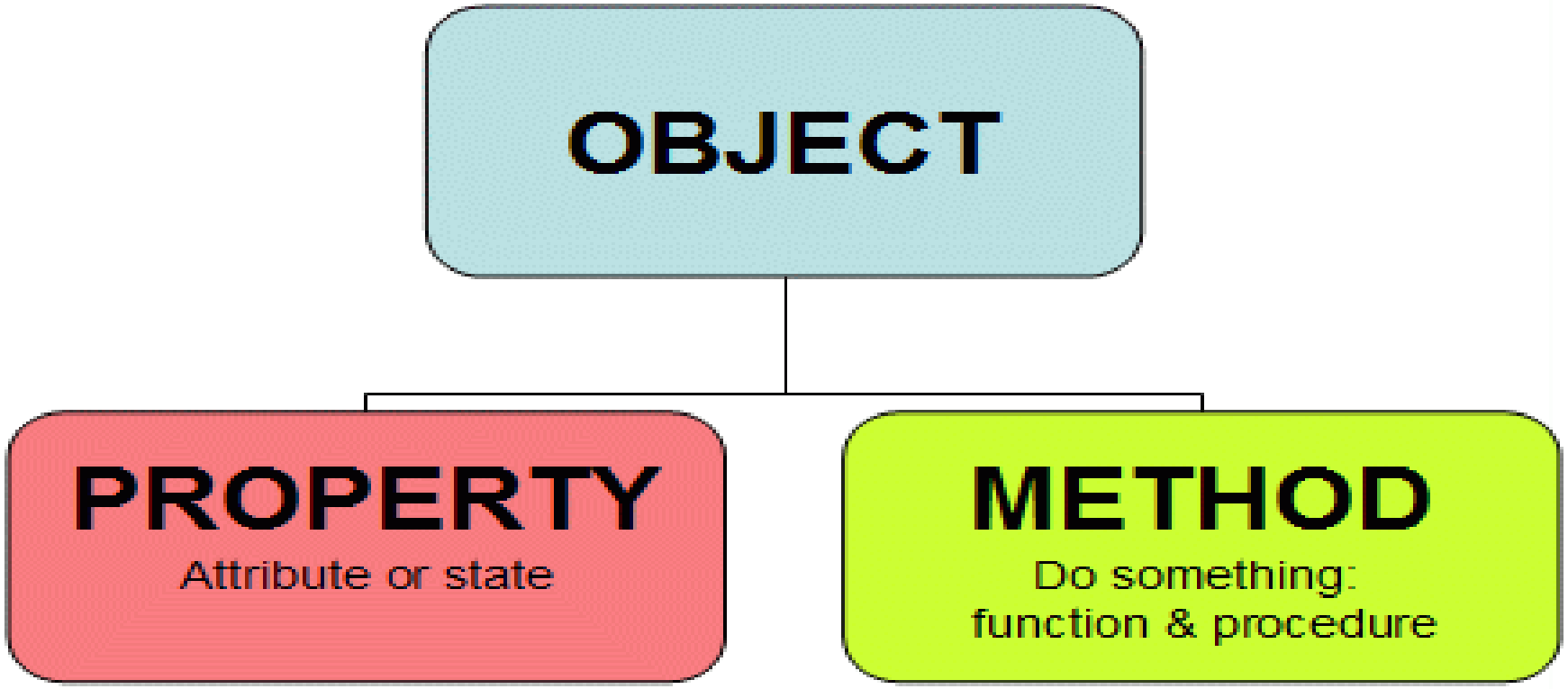
  - Note about quizes: Must do attempt 2 before midsem and 2 after midsem.
  - Textbook(s): (1) Core Java : Fundamentals, Volume I by Horstmann. Pearson (2) Core Java : Advanced Topics, Volume II by Horstmann. Pearson
  - Office hours: TBA.

Procedural Programming

- Most natural progression from algorithms to code.
- Program: set of functions, one calls another in a fixed sequence.
- Abstract information is mapped to its storage.
  - Variables are actual memory locations based on types.
- Convenient for simple procedures - best choice for code e.g. device drivers, operating system etc.

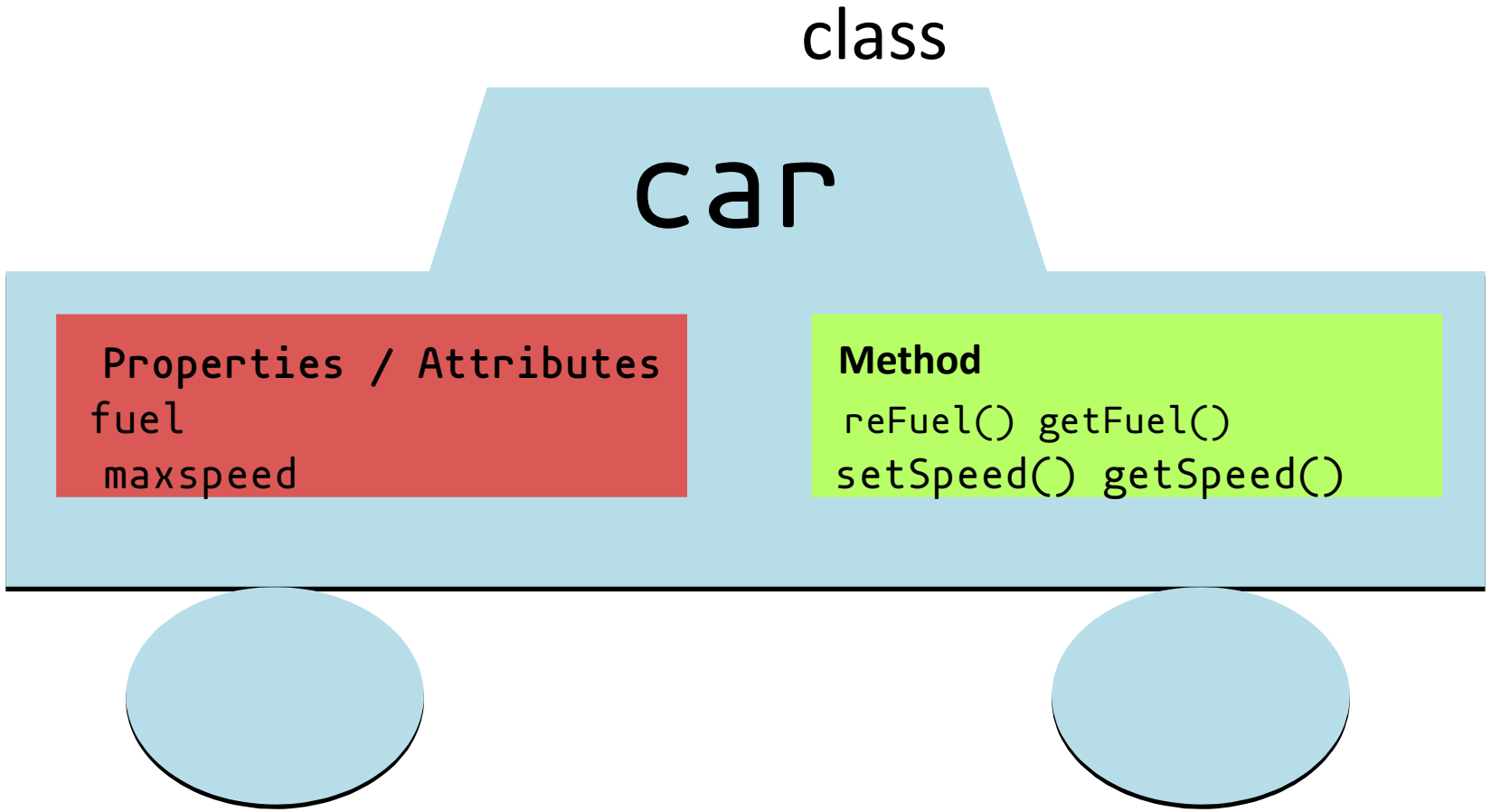
- Not convenient for large application programs where abstractions are a must.

What is OOP?



It is a programming paradigm based on the concept of “objects”, which may contain data in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods
(Wikipedia)

What is OOP?


class
car
Properties / Attributes
fuel
maxspeed
Method
reFuel() getFuel()
setSpeed() getSpeed()

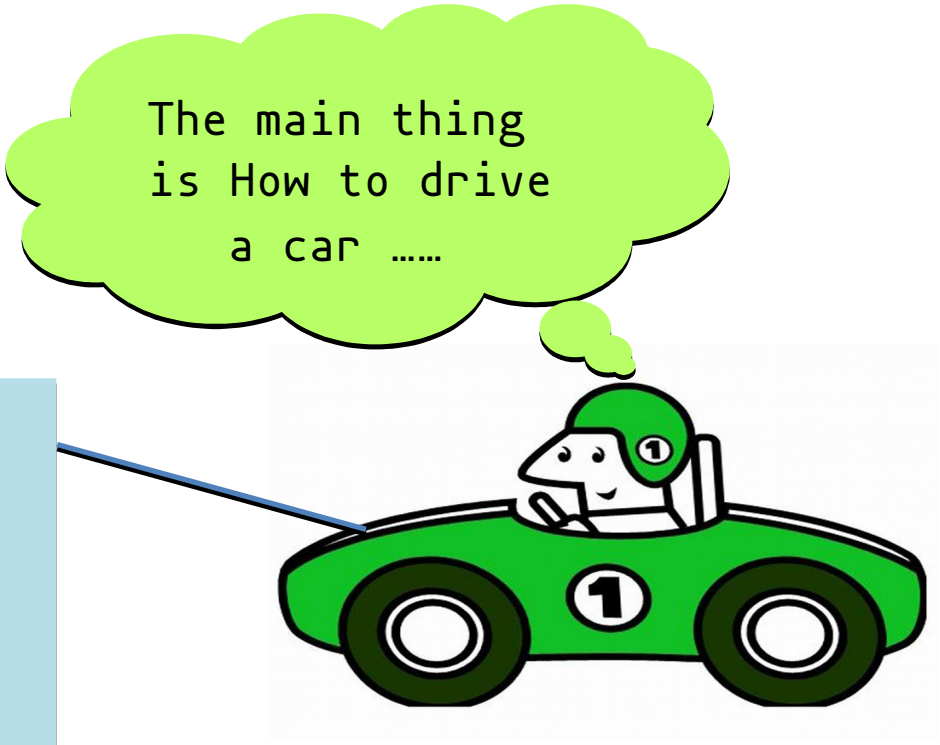
OOP Features

- **Encapsulation**
- Method overloading
- Inheritance
- Abstraction
- Method overriding
- Polymorphism

Advantages of OOP

- Code reuse and recycling
  - Objects can easily be reused
- Design benefits
  - Extensive planning phase results better design and lesser flaws
- Software maintenance
  - Easy to incorporate changes in legacy code (e.g., supporting a new hardware)
- Simplicity

Abstraction



How the car is moving and how the engine is working, this information is hidden.

(**Abstraction**)

Basics of Java Programming Language

- Installing Java on Windows/Linux/MacOS/*nix
  - Oracle/Sun Java JDK (proprietary - you only get the compiler and runtime)
  - OpenJDK/OpenJRE (open source - you can compile from the source)

Name	Acronym	Explanation
Java Development Kit	JDK	The software for programmers who want to write Java programs
Java Runtime Environment	JRE	The software for consumers who want to run Java programs
Server JRE	—	The software for running Java programs on servers
Standard Edition	SE	The Java platform for use on desktops and simple server applications
Enterprise Edition	EE	The Java platform for complex server applications
Micro Edition	ME	The Java platform for use on small devices
JavaFX	—	An alternate toolkit for graphical user interfaces that is included with certain Java SE distributions prior to Java 11
OpenJDK	—	A free and open source implementation of Java SE
Java 2	J2	An outdated term that described Java versions from 1998 until 2006
Software Development Kit	SDK	An outdated term that described the JDK from 1998 until 2006
Update	u	Oracle's term for a bug fix release up to Java 8
NetBeans	—	Oracle's integrated development environment

Basics of Java Programming Language

- My first program:

- – Everything is a “class”.
- – The file name and classname
- must be the same.
- – Welcome.java
- – Java compilation:
- $ java Welcome.java
- – Program runs in the JVM:
- $ java Welcome

- Syntax very similar to
- C/C++.

```
1  /**
2   * This program displays a greeting for the reader.
3   * @version 1.30 2014-02-27
4   * @author Cay Horstmann
5   */
6  public class Welcome
7  {
8     public static void main(String[] args)
9     {
10       String greeting = "Welcome to Core Java!";
11       System.out.println(greeting);
12       for (int i = 0; i < greeting.length(); i++)
13          System.out.print("=");
14       System.out.println();
15    }
16 }
```

Java vs C/C++

- – Machine indepenedent.
- – Runs on a Java Virtual Machine (JRE) which understands a special byte-code format.
- – Doesn’t run on bare metal.
- – High portability (Java/Python) vs performance (C/C++).

- – Is Java a compiled or an interpreted language ?

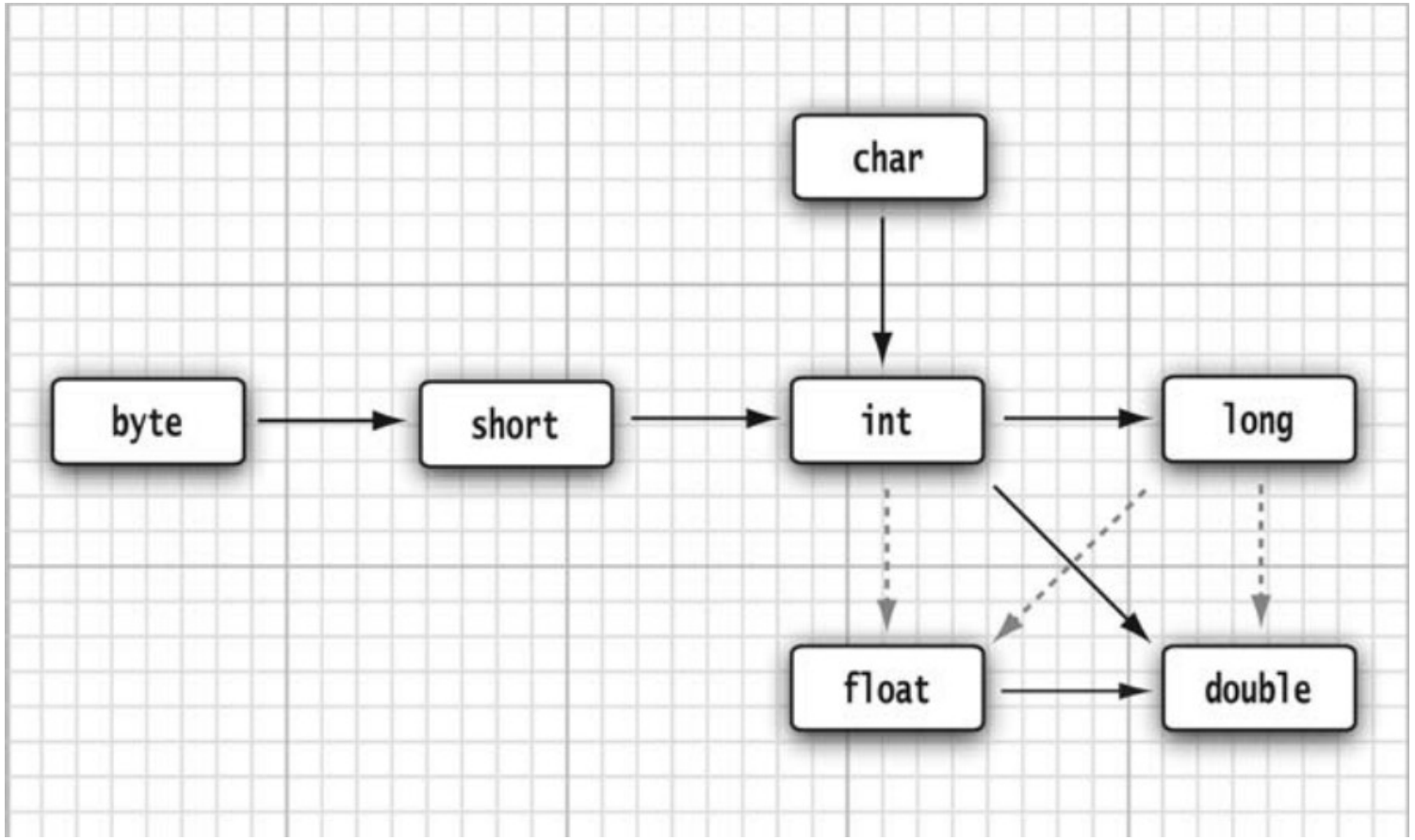
Java: Types (Integer)

Type	Storage Requirement	Range (Inclusive)
`int`	4 bytes	–2,147,483,648 to 2,147,483,647 (just over 2 billion)
`short`	2 bytes	–32,768 to 32,767
`long`	8 bytes	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
`byte`	1 byte	–128 to 127

Java: Types (Floating point)

Type	Storage Requirement	Range
`float`	4 bytes	Approximately ±3.40282347E+38F (6–7 significant decimal digits)
`double`	8 bytes	Approximately ±1.79769313486231570E+308 (15 significant decimal digits)

Java: Type casts/conversions



- Conversions without information loss
double → int [allowed by there wold be information loss]

Java: Bitwise Operators

- & (‘and’)
- | (‘or’)
- ^ (‘xor’)
- ~ (‘not’)
- << (‘left shift’)
- >> (‘right shift’)

Java: Operator Precedence and Hierarcy.

Operators	Associativity
[] . () (method call)	Left to right
! ~ ++ -- + (unary) - (unary) () (cast) new	Right to left
* / %	Left to right
+ -	Left to right
<< >> >>>	Left to right
< <= > >= instanceof	Left to right
== !=	Left to right
&	Left to right
^	Left to right
|	Left to right
&&	Left to right
||	Left to right
?:	Right to left
= += -= *= /= %= &= |= ^= <<= >>= >>>=	Right to left

Java: Abstract Data Types (Strings)

- – ‘String’ abstract type.
- – Behaves like an *immutable* C++ string.

- Substrings:
  - String greetings = “Hello”;
  - String s = greeting.substring(0,3);

- Concatenation:
- String greeting1 = “Hello”;
- String greeting2 = “World”;
- String message = greeting1 + greeting2;

- Testing string equality:
- equals() method of String class used for testing equality of two strings.
- <string object1>.equals(<string object2>)
- Output: boolean (true or false).

String Builder

```
String builder = new StringBuilder();
builder.append(ch);
builder.append(str);
String longString = builder.toString();
```

Basic I/O

- Read input from stdin and write to stdout.
- Class ‘Scanner’

```
import java.util.*;

Scanner in = new Scanner(System.in);

in.next();
in.nextLine();
in.nextInt();
in.nextDouble();
in.hasNextInt() <-- boolean
in.hasNextDouble() <-- boolean
```

Basic I/O

- Read input from stdin and write to stdout.
- Class ‘Scanner’

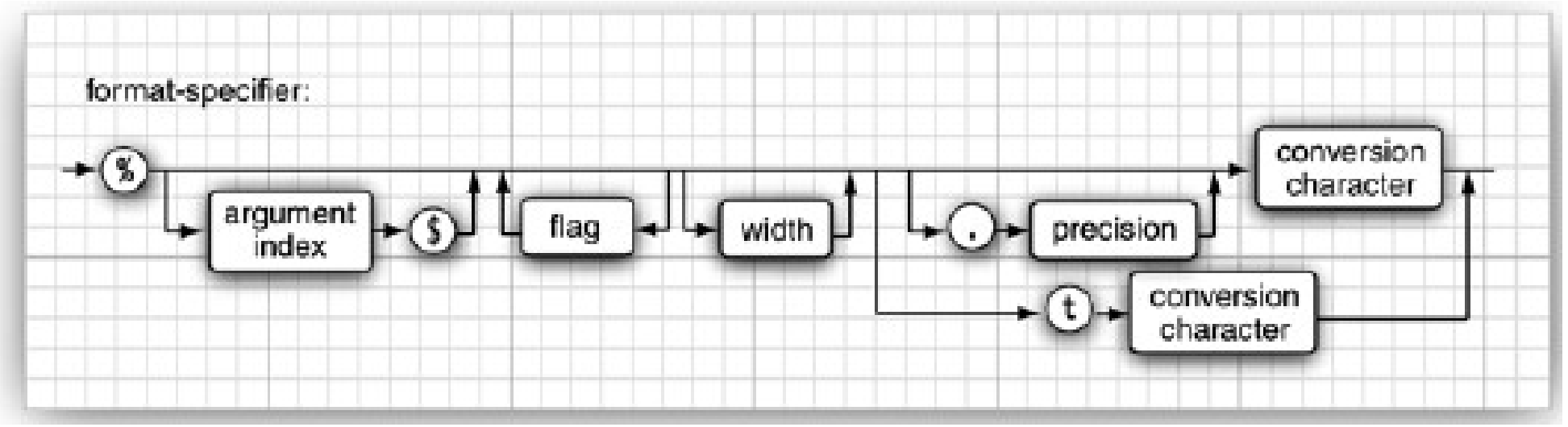
```
import java.util.*;

Scanner in = new Scanner(System.in);

in.next();
in.nextLine();
in.nextInt();
in.nextDouble();
in.hasNextInt() <-- boolean
in.hasNextDouble() <-- boolean
```

System.out.printf() <-- Java adaptation to C/C++ stdio printf()

Basic I/O

Conversion Character	Type	Example
d	Decimal integer	159
x	Hexadecimal integer	9f
o	Octal integer	237
f	Fixed-point floating-point	15.9
e	Exponential floating-point	1.59e+01
g	General floating-point (the shorter of e and f)	—
a	Hexadecimal floating-point	0x1.fccdp3
s	String	Hello
c	Character	H
b	boolean	true
h	Hash code	42628b2
t*x* or T*x*	Date and time (T forces uppercase)	Obsolete, use the java.time classes instead—see Chapter 6 of Volume II
%	The percent symbol	%
n	The platform-dependent line separator	—

Basic I/O (printf format specifier)



Control Flows

- if else
- while()
- for()
- switch case
- do{ }while()
- break and continue

Semantics same as C/C++

Scope

– Determines the point in the code upto which a variable/function name etc. are valid.

- – Global / local

- – Also associated with functions inside a class.
- – Java is more conservative about scopes than C/C++.

```
public static void main(String[] args)
{
   int n;
   . . .
   {
      int k;
      . . .
   } // k is only defined up to here
}
```

```
public static void main(String[] args)
{
   int n;
   {
      int k;
      int n; // ERROR--can't redefine n in inner block
      . . .
   }
}
```

Arrays and References

- – Contiguous ``allocate memory''
- int[] arr = **new** int[100]; //**new** keyword mandatory.

- – Reference: much like pointers in C/C++ – Name of the object or array.
- int[] arr = **new** int[100];
- int[] myref;
- myref = arr; // myref → arr : reference.

Multidimensional Arrays

- <type>[][] <var-name>;

- int[][] marray;
- marray = new int [100];

- for (int i=0;i<100;i++){
- marray[i] = new int[100];
- …
- }