Instructor: Ken Brumer

Advanced Programming Techniques in Java

COSI 12B

COSI 12b

- **Class**
  - Tu, Th 2.20pm-3.40pm, online
  - recordings, lecture notes, syllabus, contact info on LATTE
  - Recitation: Tu 5.30-6:50pm, online
- **Office hours**
  - MF 9am-10am
- **TAs**
  - TAs: introduced on LATTE, and recitation

Your Instructor

- **Ken Brumer**
  - Pronouns: he/him
  - Programming professionally for over 20 years
  - Started programming in BASIC (1987)
  - Email: kenbrumer@brandeis.edu

Important Points

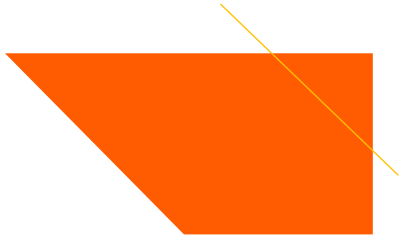
- At any point, ask me WHY?
- You can ask me anything about the course in class, during a break, on my office hours, by email

What is COSI 12b about?

- Teaches Object Oriented Programming (OOP)
  - OOP Basics
  - Definition, creation and usage of classes, objects and methods
- Advanced topics
  - Inheritance, polymorphism, interfaces, Java Collection Framework, Java API
- Provides the foundations of good programming skills

Object-Oriented Programming (OOP)

- OOP is now the dominant way to program, yet it is over 40 years old
  - Alan Kay received in 2003 the ACM's Turing Award for Smaltalk the first OOPL
- OOP was slow to catch on, but since the mid-90's everybody is been using it
- OOP emphasizes objects, which often reflect real-life objects
  - Have both properties and capabilities (i.e., they can perform tasks: "they know how to …")



OOP as Modeling

- In OOP, a program is modeled as a collection of cooperating objects
- Objects are “smart” in their specialty
  - e.g., bed can make itself, door can open itself, menu can let selections be picked
  - But, each must be told when to perform actions

OOP as Modeling

- In OOP, a program is modeled as a collection of cooperating objects
- Objects are “smart” in their specialty
  - e.g., bed can make itself, door can open itself, menu can let selections be picked
  - But, each must be told when to perform actions

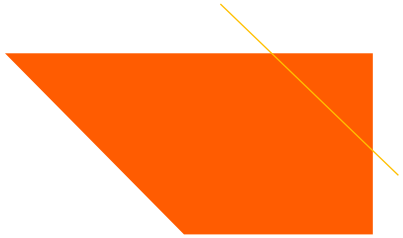
How many Java programmers does it take to change a light bulb?

OOP as Modeling

- In OOP, a program is modeled as a collection of cooperating objects
- Objects are “smart” in their specialty
  - e.g., bed can make itself, door can open itself, menu can let selections be picked
  - But, each must be told when to perform actions

How many Java programmers does it take to change a light bulb?

None, you just send it a message, and it changes itself



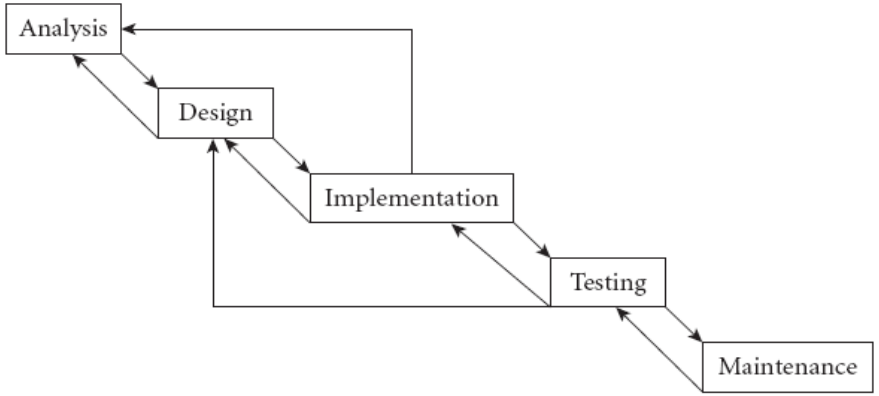
OOP as Modeling

- In OOP, a program is modeled as a collection of cooperating objects
- Objects are “smart” in their specialty
  - e.g., bed can make itself, door can open itself, menu can let selections be picked
  - But, each must be told when to perform actions
- Each object represents an abstraction
  - A “black box” hides details you do not care about
  - Allows you as the programmer to control program’s complexity
- We will write programs by modeling problems as set of collaborating components

Why do we like Java?

- Syntax similar to C++ but simpler, cleaner and more beginner-friendly
  - e.g., Java handles memory management for you
- It allows platform independence
  - Write once run anywhere (WORA)
  - Windows, Mac, Linux, cell phones
- Cuts development and debugging time
- Extensive libraries available
- It is the most prevalent language for the Web and in industry today

Software Development



- Analysis
  - English description of what the system models to meet user requirement/ specification
- Designing the system
    - System is composed of smaller subsystems which in turn may be composed of even smaller subsystems (diagrams often helpful)
- Implementing the design
    - If design is good, most of the hard work should be done
- Testing and Debugging
    - Testing: submitting input data or sample user interactions and seeing if program reacts properly
    - Debugging: Process of removing program bugs (errors)
- Maintenance

What is COSI 12b about?

- Teaches Object Oriented Programming (OOP)
  - OOP Basics
  - Definition, creation and usage of classes, objects and methods
- Advanced topics
  - Inheritance, polymorphism, interfaces, Java Collection Framework, Java API, recursion
- Provides the foundations of good programming skills

Who is it for?

- Prospective CS majors/minors
- Students with varying levels of programming experience
  - C/C++, Python (need to take placement test)
  - Java fundamentals (need to take placement test)
    - Scalar types: integers, strings, booleans
    - Control structures: loops, conditional expressions, assignment statements
  - COSI 10a
- Anyone who wants an in-depth introduction to OO programming

Placement Exam

- You are required to take the placement exam to determine if you can enroll in COSI 12b
- The exam will be available until the end of the 2nd day of class.
  - If you have taken the AP Computer Science A exam and received 5 you don’t need to take the test
- This exam test your knowledge of control flow, ability to trace code, primitive data types vs reference types. You will need to demonstrate good knowledge of String, Math classes, familiarity with fundamental concepts of classes and objects, and fluency working with arrays

https://www.brandeis.edu/computer-science/undergraduate/placement.html

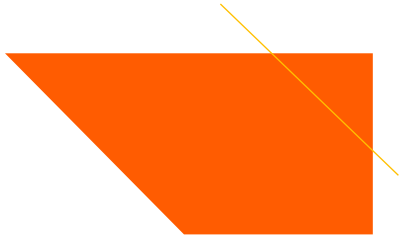
Course Mechanics

- **Resources**
  - Lecture Notes
  - Textbook: : Building Java Programs - A Back to Basics Approach (4th Edition) by Stuart Reges and Marty Stepp, Addison Wesley

- **Software**
  - The recommended software for the course is the last version (17) of the Java Development Kit (JDK), and
  - the most recent version of Java IDE (Eclipse)

Course Mechanics: Latte page

- You will find (everything):
  - Syllabus
  - Lecture slides (you need them for your HWs, especially if you don't have the book)
  - Recordings
  - Programming Assignments
  - Support code
  - Tutorial videos
  - Submit your assignment
  - You are responsible for everything there
- **Questions? Discussion forum on Latte**
  - E.g. [PA1] – Output format of problem 1

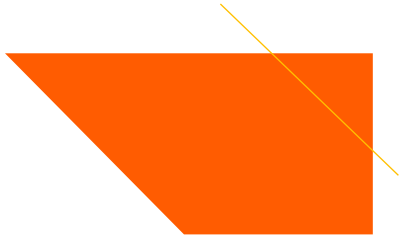


Course Mechanics: Questions!

1. For general questions on the programming assignments requirements, clarifications etc the questions should be asked on the discussion forum first.
2. For issues you may be having with your code, and individual questions that can not be asked at the discussion forum, you should go to the TAs office hours, or talk to the TAs during recitation.
3. For general questions on the topics of the course, you should go to my office hours, or the TAs office hours.

Course Mechanics: Course Workload

- 6 Programming Assignments (roughly every other week)
- Two Exams: **Mar 7**, **May 2**
  - Exams during class time
- Your final grade for the course will be determined using the following weights:
  - Homework: 50%
  - Exams: 50%
- Make-up exams will not be given except in case of a serious emergency



Course Mechanics: Homework

- Homework consists of bi-weekly programming assignments done individually and submitted electronically via Latte
- Programs will be graded on:
  - “external correctness” (behavior)
  - “internal correctness” (style and design)
- **Regrading**:
  - Any re-grading must be submitted to via Latte (by filling the form) at most **four days** after grades are released
  - Any re-grading requests after this period will not be granted

Course Mechanics: Late policy

- Three free late days that you can use on assignments throughout the semester
  - Each day used extends a deadline by 24 hours with no penalty
  - No need to email us, these will be applied automatically when you submit late.
  - **Once a student has used up all the late days**, late submissions penalties will be applied based on the syllabus. Submissions will not be accepted a week (7 days or more) after the deadline.
  - Check syllabus for penalty per late day!

Course Mechanics: Recitation

- Every Tuesday from 5.30-6.50pm
- Online

Know what is and isn't legal

- As a student of this course, you are agreeing to the following principles:
  - When there is doubt regarding the honorability of an action, you will ask before doing it
  - When possible, to do so with honor, you will help your fellows to learn and improve
  - You will get help from course staff before succumbing to frustration. **Frustration leads to the dark path.**
  - You may discuss general ideas of how to approach an assignment, but never specific details about the code to write. Any help you receive from or provide to classmates should be limited and should never involve details of how to code a solution

Key to Success

- Start early
- Work steadily
- Don’t fall behind

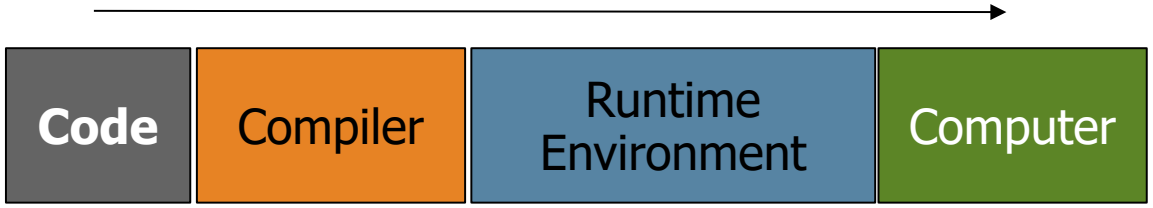
Questions?

- About the course.

Java vs Python Language

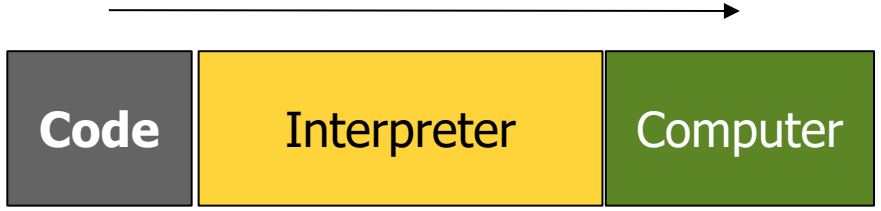
- Compiled language

Java:



- Interpreted language
- Code is written and then directly executed by an **interpreter**
- Type commands into interpreter and see immediate results

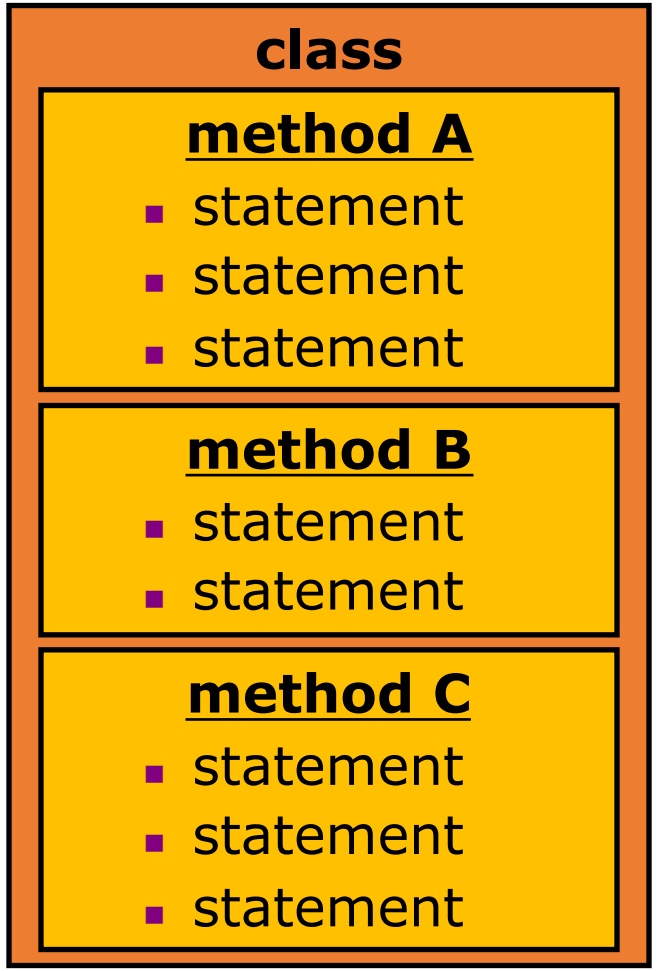
Python:



Java Program Structure

- In the Java programming language:
  - A program is made up of one or more *classes*
  - A class contains one or more *methods*
  - A method contains program *statements*

- A Java application always contains a method called `main`



Java Program Structure

```
//  comments about the class
public class MyProgram
{


}
```

class header

class body

Comments can be placed almost anywhere

Java Program Structure

```
//  comments about the class
public class MyProgram
{

   //  comments about the method
   public static void main (String[] args)
   {

   }

}
```

Comments

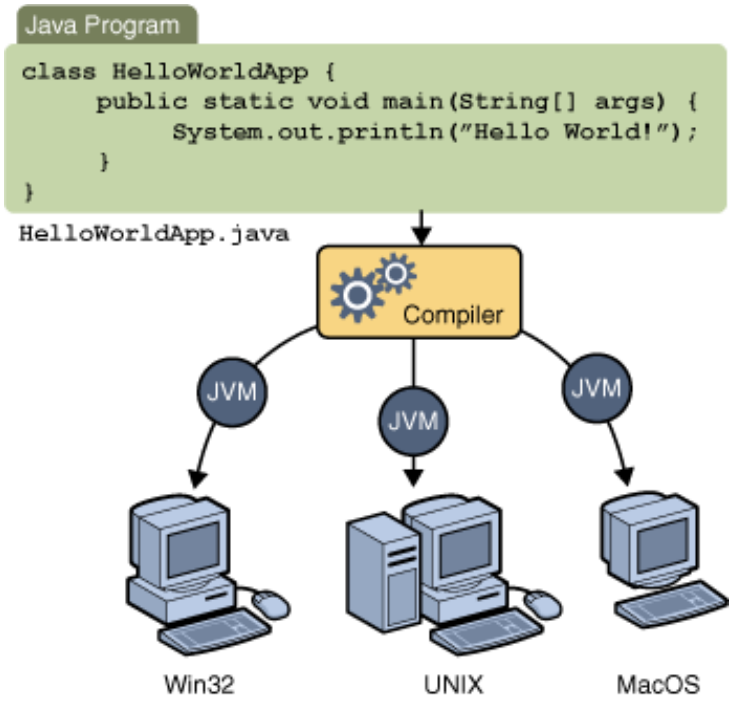
- Comments in a program are called *inline documentation*
- They should be included to explain the purpose of the program and describe processing steps
- They do not affect how a program works
- Java comments can take three forms:

```
// this comment runs to the end of the line

/*  this comment runs to the terminating
    symbol, even across line breaks        */

/** this is a javadoc comment   */
```

Writing, Compiling and Executing a Java Program



- **Step 1: You write your code**
- **Step 2: You compile your code** `javac HelloWorldApp.java`
  - The Java compiler translates Java source code into a special representation called *bytecode*
  - Java bytecode is not the machine language for any traditional CPU
- **Step 3: You run your code** `java HelloWorldApp`
  - An *interpreter*, translates bytecode into machine language in the Java Runtime Environment (JRE) and executes it
  - JRE is specific to your platform
  - JRE contains class libraries which are loaded at runtime
  - The JRE creates a virtual machine within your computer known as the JVM (Java Virtual Machine). This is the reason Java is considered to be *architecture-neutral*

So far …

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

- Everything in Java must be inside a class
- Every file may only contain one public class
- The name of the file must be the name of the class appended to the **`java`** extension

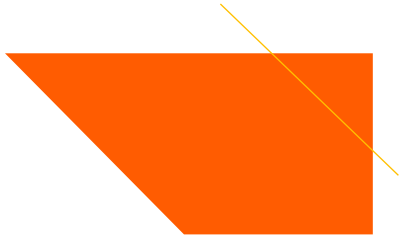
- Thus, **`Hello.java`** must contain one public class named **`Hello`**

System.out.println

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

- It's a *statement* that prints a line of output on the console

- Two ways to use **`System.out.println`**
  - `System.out.println(); //`Prints a blank line of output
  - `System.out.println("<text>"); //`Prints the text message as output



println vs. print

- **println** sends output to the current line, and then it advances to the next line
- **print** sends output to the current line without advancing to the next line, Therefore anything printed after a `print` statement will appear on the same line

```
System.out.print("To be ");
System.out.print("or not to be. ");
System.out.print("That is ");
System.out.println("the question.");
System.out.print("This is");
System.out.println(" for the whole family!");
```

```
To be or not to be. That is the question.
This is  for the whole family!
^
```

Formatting text with `printf()`

- Syntax

```
System.out.printf("format string", <list parameters>);
```

- The *format string* is like placeholders where the parameters are inserted
  - These placeholders are used instead of + concatenation
    - %d integer
    - %f real numbers
    - %s string
- Example

```
int x = 3;
int y = -17;
System.out.printf("x is %d and y is %d\n", x, y);
```

Note: `printf()` does not drop to the next line unless you use `\n`

printf precision

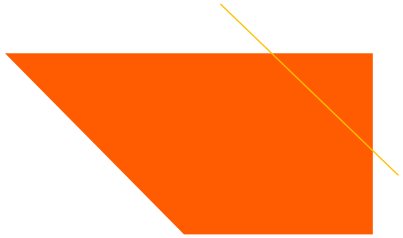
%.Df real number, rounded to **D** digits after decimal

%W.Df real number, **W** characters wide, **D** digits after decimal

```
double gpa = 3.253764;
System.out.printf("your GPA is %.1f\n", gpa);
System.out.printf("more precisely: %8.3f\n", gpa);
```

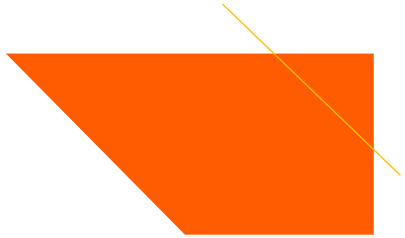
Output

```
your GPA is 3.3
more precisely:    3.254
```



`printf` with `Strings`

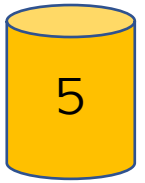
A simple string	`printf("'%s'", "Hello");`	`'Hello'`
A string with a minimum length	`printf("'%10s'", "Hello");`	`'     Hello'`
Minimum length, left-justified	`printf("'%-10s'", "Hello");`	`'Hello     '`



Variables and Data Types

Variables

- A **variable** is a name for a location in memory
  - It can be thought of as a container which holds values for you



- A variable must be declared by specifying the variable's name and the type of information that it will hold

```
int total;
```

data type

variable name

Variables

- In order to use a variable in a program you to need to perform 2 steps:
  - Variable Declaration
  - Variable Initialization

- A variable can be given an initial value in the declaration

```
int total = 0;
```

Assignment

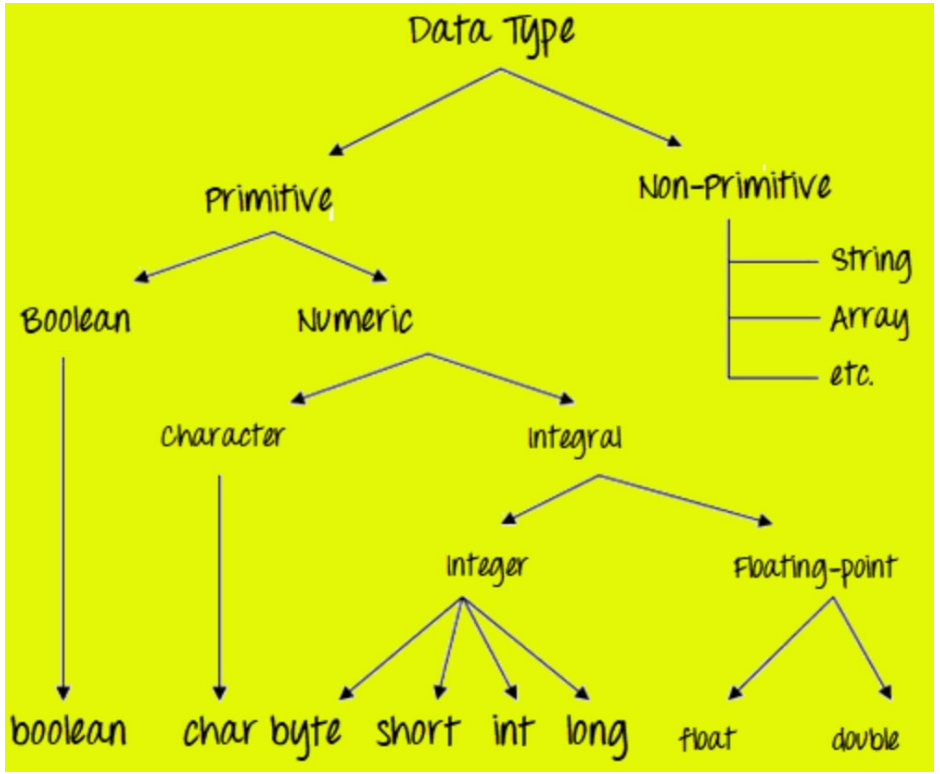
- An *assignment statement* changes the value of a variable

```
total = 35;
```

- The value that was originally in `total` is overwritten
- You can assign only a value to a variable that is consistent with the variable's declared type

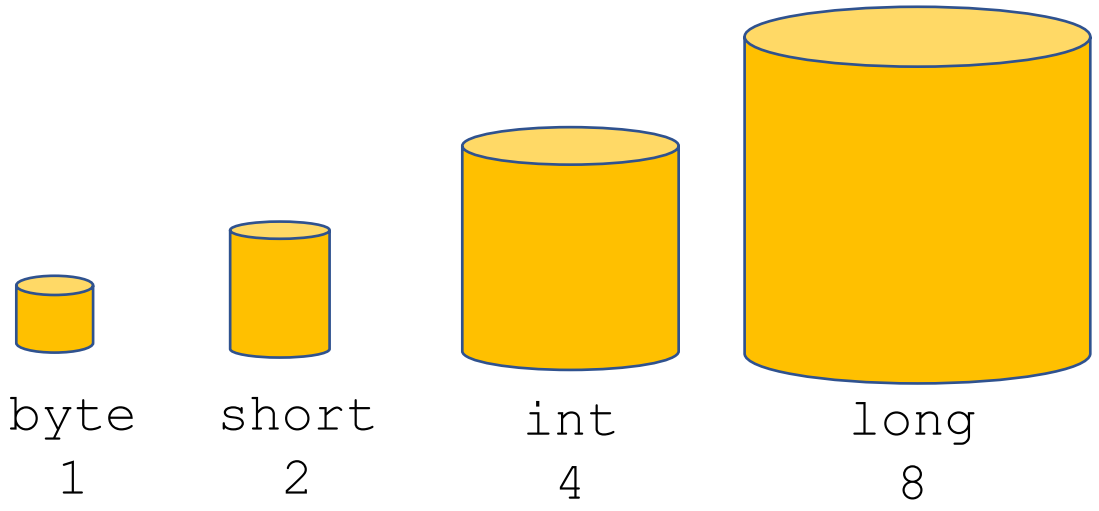
Data Types

- Data types classify the different values to be stored in the variable
- In Java there are two types of data types:
  - Primitive Data Types
  - Non-primitive Data Types

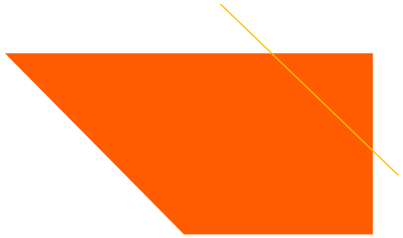


Primitive Data Types

- Primitive Data Types are predefined and available within the Java language
- There are 8 primitive types: `byte, short, int, long, char, float, double, and boolean`



Data type	Default Value	Default size
byte	0	1 byte
short	0	2 bytes
int	0	4 bytes
long		8 bytes
float	0.0f	4 bytes
double	0.0d	8 bytes
boolean	false	1 bit
char	'\u0000'	2 bytes



Primitive Data Types

- byte -128 to 127
- short -32,768 to 32,767
- int -2,147,483,648 to 2,147,483,647
- long -9,223,372,036,854,775,808 to …
- float $\pm 10^{38}$ incl. 0 with 6 digits of precision
- double $\pm 10^{308}$ incl. 0 with 15 digits of precision
- char Unicode character set
- boolean true, false

Example

```
public class ChangeAdder {
	public static void main(String[] args){
		int quarters = 10;
		int dimes = 3;
		int nickels = 7;
		int pennies = 6;
		int change = 0;
		change = 25*quarters+10*dimes+5*nickels+pennies;
		System.out.println("total in change is:" + change);
	}
}
```