



CS 2100: Data Structures & Algorithms I

Hello World; Primitive Data Types;
Using Simple Objects

Dr. Nada Basit // [basit\[at\]Virginia\[dot\]edu](mailto:basit[at]Virginia[dot]edu)

Spring 2022

Friendly Reminders

- Masks are **required** at all times during class (University Policy)
- If you forget your mask (or mask is lost/broken), I have a few available
 - **Just come up to me at the start of class and ask!**
- No eating or drinking in the classroom, please
- Our lectures will be **recorded** (see Collab) – please allow 24-48 hrs to post
- If you feel **unwell**, or think you are, **please stay home**
 - *We will work with you!*
 - At home: eye mask instead! **Get some rest** 😊



Reminder of my Contact Information

- **Dr. Nada Basit**

- Office: rice Hall 405
- OH: Mon (1:15-2:15pm) and Tue (11:30-1:00pm) on *Zoom*
- basit@virginia.edu

Best way to get in touch with me!

(Always include "CS 2100" in email subject line)

**Prof. Basit's
Office Hours Challenge!**



Introduction to Java

A Little About Java

- Java is a powerful and popular programming language
- It's broadly used because it is:
 - **Portable**
 - Write once, run anywhere...
 - Desktop software
 - Android apps
 - Enterprise systems
 - **Safe**
 - Based on C syntax, common C issues addressed by the compiler
 - **Object-Oriented** (as opposed to procedural)

A Little About Java

- Java is **Object-Oriented**
 - Different **paradigm** for programming
 - Everything is an “**object**”
 - Objects may *contain other objects*
 - Objects have a type of class
(defines what you can ask of the object)



A Little About Java

- Java is **statically** typed
 - All variables must be **declared** before they are used
 - Must provide a **type** for all variables
 - int, long, float, double, String, **Cat**, ...

- For example:
type followed by the variable *name*, such as:

```
int numOfPages = 312;
```

- Java is **strongly** typed
 - Strongly typed languages force the types stored in variables to be what is **expected**
 - A variable will **not** automatically be converted from one type to another



Java: Some (naming) Conventions

- Case sensitive:
 - Classes are **T**itleCase:
 - Variables are **c**amelCase:
 - **Constants**: variables whose value is not permitted to change:
 - Coding convention: ALL CAPS
 - Use “**final**” keyword to indicate a **constant**
 - Example: **final int CAR_NUM_WHEELS = 4;**
-
- **Comments**: use // instead of #
 // Java compiler ignores this line!

Example

unicorn != **U**nicorn

My**F**irst**C**lass

rectangle**W**idth



A Little About How Java Works

- Java and Javac
- **Javac is the Java Compiler**
- You write: MyProgram.java
 Compiler: MyProgram.class
- **Java reads and executes .class files (not human-readable)**
- Your file, MyProgram.java, must contain a **class name** MyProgram
- If your file does not contain a “**main**” method, it will not do anything...!
 - You **do not** *have* to have a main method in your files unless you want it to do something!

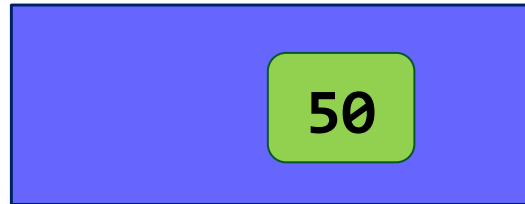
• **When submitting homework: ONLY SUBMIT . JAVA FILES!** 😊



Variable

- A variable is simply a name associated with a specific object or primitive data

RAM:



- A name associated with a **reserved area** allocated in **memory**
- Has a type – Java is *statically* typed (**variables declared before they're used**)
 - Can be **primitive** or **reference** (*more on this later*)

```
int num = 50;
```

Static

(no not that kind...)



- What does static mean?
 - Anything *static* is accessible without an object of the class
 - (Accessed / “called” directly)
- The **main** method is needed to run things in your program, and it is a *static* method!

```
public static void main (String[] args) { ...
```

Hello World in Java

```
import java.io.*;

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

This is a simple yet complete Java program. It does one thing: Prints “**Hello World!**”

Output:

Hello World!

Hello World in Java (with Comments)

```
/* Below is an import statement
 * it is used if you want to use code from other packages ←
 */
/* Java.io.* is all of Java's input/output stuff */
import java.io.*;

public class HelloWorld { // Class declaration (common single-line comment)
    /**
     * The main method of the program.
     * This is a Java doc comment, note the " /** "
     * @param args - variable for the input array of Strings
     */
    public static void main (String[] args) {
        /* This is how you print to the console */
        System.out.println("Hello World!");
    }
}
```

(This is a multi-line comment, note the " /* ")

←

What is System.out.println() ???

Notes on `System.out.println()`;

```
import java.io.*;
```

```
public class HelloWorld {  
    public static void main (String[] args) {
```

```
        /* if you use println, Java puts a new line at the end */  
        System.out.println("Hello");
```

```
        /* And starts the next output on a new line */  
        System.out.println("There.");
```

```
        /* If you use just print(), no newline after printing */  
        System.out.print("How are "); // remember to add a space!  
        System.out.print("you?");
```

```
    }  
}
```

Remember, if you write a program in **main** but have **no output statements** then you will **see no output in the console**. It doesn't mean nothing happened!

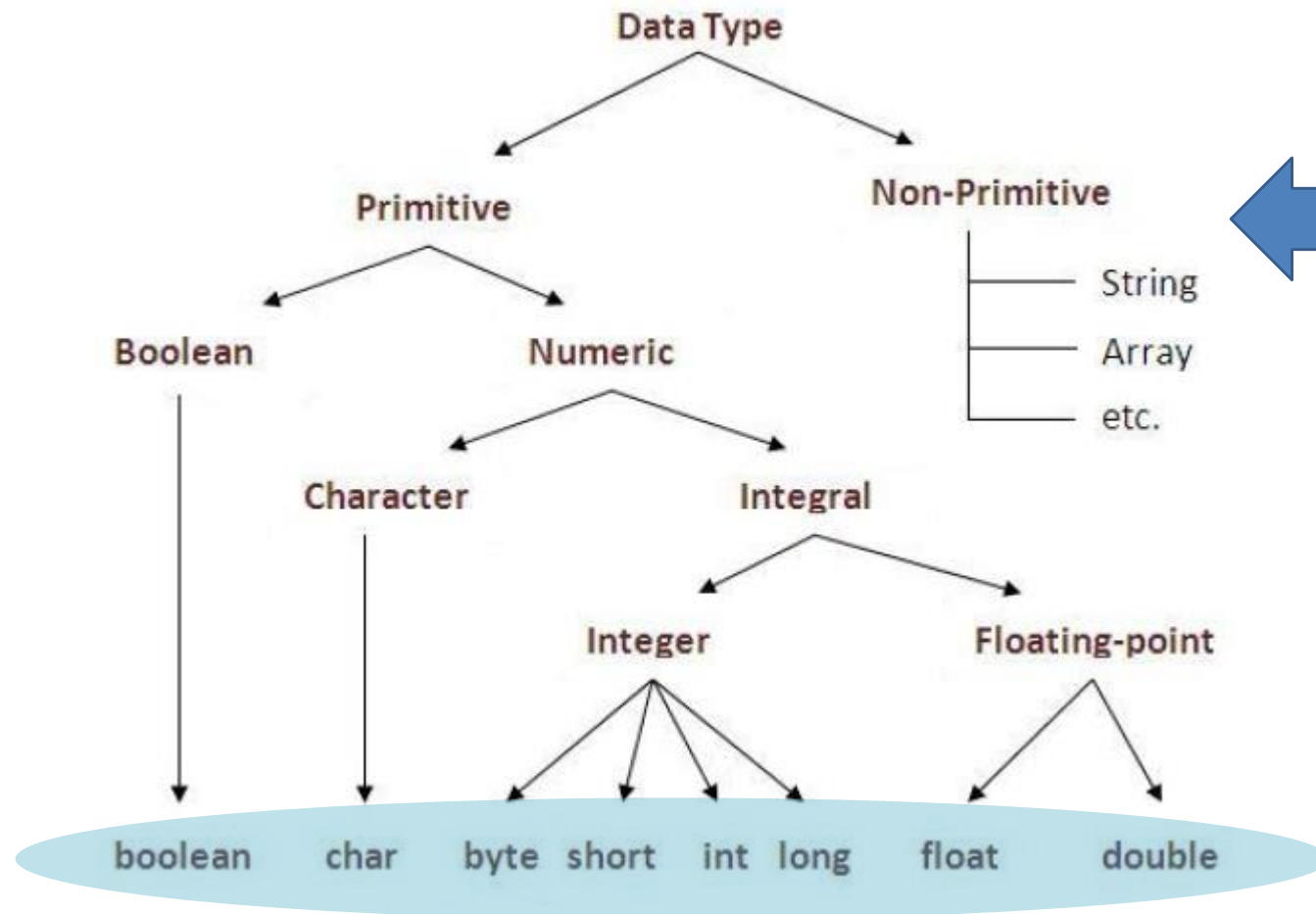


Output:

```
Hello  
There.  
How are you?
```

Data Types & Objects

Java Data Types (primitive and non-primitive)



Reference Types and String (the strange one!)

Eight (8) Primitive Data Types in Java

Data Types

- Data type determines:

- What kind of **information** the variable holds
- What **operations** can you do on that information?



- **Boolean**

- True, False
- → AND, OR, NOT...

- Does “*multiply*” make sense? **NO!**

- **Integer**

- (+/-) whole numbers, zero
- → Add, subtract, divide, multiply...

- Does “*NOT*” make sense? **NO!**

Primitive Data Types

For each data type the **space allocated in memory** is shown (in bytes).


The **variable name** is used to access this memory (e.g. **var1**).

```
/* What you see below is ALL of Java's primitive types */
```

```
/* various types of integers */
```

```
byte  var1  = 10;    //1 byte [-128, 127]
short var2  = 12;    //2 bytes [-32768, 32767]
int   var3  = 90;    //4 bytes [-2^31, 2^31 - 1]
long  var4  = 14;    //8 bytes [-2^63, 2^63 - 1]
```

Numerical range shown
[lowerbound, upperbound]



```
/* various floating point types */
```

```
float  fp1 = 1.34f; //4 byte IEEE 754 number
double fp2 = 711.2; //8 byte IEEE 754 number
```

Note: lower case
"true" and "false"



```
/* other */
```

```
boolean b = true;    //can have values true and false
char     c = 'd';    //2 bytes, stores a single character
```

Primitive Data Types

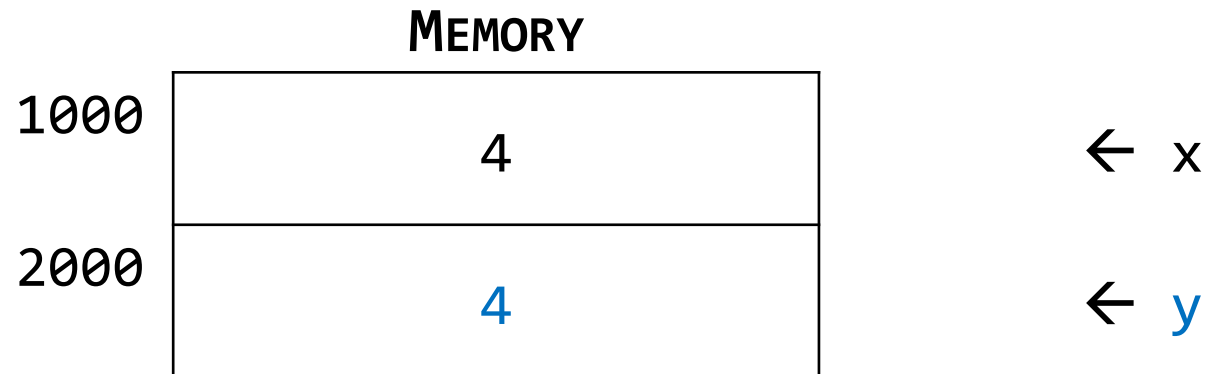
Primitive Types:

(“non-reference types”)

```
int x = 4;
```

```
int y = x;
```

Two copies of the data made



Actual values are stored in memory.

Objects: “Random” example

Java provides a random number generator

In this file we will import
java.util.Random

All other variables in Java are objects!

```
/* Strings in Java are objects */
```

```
String s1 = "Hi There";
```

```
/* Many other objects exist, here's one example */
```

```
/* Note new variables use new keyword to create */
```

```
// *** Random Example - example of reference type
```

```
Random randomGenerator = new Random(); // using key-word new
```

```
int randomInt = randomGenerator.nextInt(25); // Random number 0-24 (calling method nextInt())
```

```
// .nextInt(n) means generating a random number between 0 (inclusive) and n (exclusive).
```

```
System.out.println(randomInt);
```

```
System.out.println("-----");
```

nextInt() method exists in the Random class:
java.util.Random

Output: (one run)

14

Some Notes on Objects

- All other variables in Java are **Objects**
 - Object data allocated in memory, variable name is a reference to it (more later)
 - **Objects contain fields and methods (more coming soon)**
 - You can develop your own Objects (*aaaagain*, more on this soon!)
 - The **Java API** can tell you all of the objects Java has built in (very useful!)
 - <https://docs.oracle.com/javase/10/docs/api/overview-summary.html#JavaSE>

Brief: Primitive vs. Reference Data Types

- **Primitive data types** (*built into* Java)
 - “Literal” values, refers to literal value on **stack**
 - A “box” or chunk of memory holding the *value* itself
 - May be compared with double equal sign, `a == b`
- **Reference types** (defined from *classes*)
 - The “object” refers to the chunk of memory that holds the data
 - **The variable “points to” the object in memory**
 - Create new chunks (on **heap**) with **new** keyword
 - Calls a **constructor** for that class
 - Must be compared with special `.equals()` method. Why?



Declaring Variables

- **Declaring variables** is an **assignment** statement
 - Copy the right side to the left side
 - `int x = 4;`
 - Create space for an integer, name it 'x', and put the number 4 in that space
 - **Primitive** variables create space on the **stack** (at *compile* time)
`float radius = 1.246`
 - **Reference** variables use space on the **heap** (at *run* time)
`Cat freckles = new Cat("Mr. Freckles");`

Objects: “Scanner” example

Java provides a set of object types for reading input from the user

```
import java.util.Scanner;
```



```
public class InputExample {  
    public static void main(String args[]) {  
        System.out.println("Enter two numbers: ");  
        Scanner in = new Scanner(System.in);  
  
        int x1 = in.nextInt(); // reading in 1st number entered by user  
        int x2 = in.nextInt(); // reading in 2nd number entered by user  
        if(x1 > x2) System.out.println("First one is bigger!");  
        else if(x2 > x1) System.out.println("Second is bigger!");  
        else System.out.println("They are the same!");  
        /* Scanner also contains nextFloat(), nextLine(), etc. for other types */  
    }  
}
```

Output: (entering #s in)

```
Enter two numbers:  
24 66  
Second is bigger!
```


Methods

- `nextInt()` (see code snippet below) is called a **method**
 - Like a function, but operates on a specific instance of that type
 - In this case, get the `nextInt()` specifically from the object 'in'
 - How do you know what methods are available?
 - See the **Java API** !!
- More methods coming soon!

```
int x1 = in.nextInt(); // From the previous slide
```

Reminder... Java is Strongly Typed

- Variables in Python are NOT strongly typed. I can reassign a double to a variable that was a String:

```
x = "hello there"  
x = 5.2
```

- In Java, most of these are invalid:

```
int x = 5.23;    //Error: cannot convert from double to int  
String s1 = 9;  //Error: cannot convert from int to String  
double d1 = 5.2 //This one looks fine  
d1 = 'e'        //Error: cannot convert from char to double
```

Primitive (numeric) Data Type ranking (High→Low)

- double, float, long, int, short, byte

HIGH

LOW

----- need to cast ----->

<----- no casting -----

Reminder of ranking [high to low]:
double, float, long, int, short, byte

Casting (conversion between types)

```
// When converting Higher --> Low, need to cast the type
// double is a higher ranking than int
double d = 208.4;
int i = 2;
//int res = d / i; // error - Java won't automatically cast (High -> Low)
[uncomment this line to see error]
double res = d / i; // now this is OK
System.out.println(res); // OUTPUT: 104.2
int res2 = (int) (d / i); // Need to cast due to information loss
System.out.println(res2); // OUTPUT: 104
System.out.println("-----");
String s1 = (String)9; // Error cannot cast from int into to String
// Sometimes Java does not know how to force the conversion
```

Reminders



Syllabus Quiz

- ***Mandatory!*** Take by **Jan. 28 @ 11:59pm**. Must get **100%** to stay in the course! May take it as many times as needed. *Take it early! (Located on Collab)*

Regrades

- Request within **7 days** for hand-graded assignments

Academic Integrity

- Collaboration: discuss within your **cohort** but do your own work; **single source** at a time; ability to **explain**

Deadlines are at **11:59pm ET!**



Quick & Fun Survey Questions

Got any Toggle Questions you would like me to ask the class? If so, send me email and I'll ask in class next time!



**Be an Active Participant in
Your Learning!
Be Curious!
Ask Questions!**