PRIMITIVE DATA & VARIABLES

Dr. Nazli Hardy, MBA, PhD

adapted from "Building Java Programs", Reges, Stepp

OVERVIEW

- Introduction
- Primitive Data Types
- Expressions
- Literals
- Arithmetic Operators
- Mixing Types & Casting
- Precedence
- Variables
- Declaration

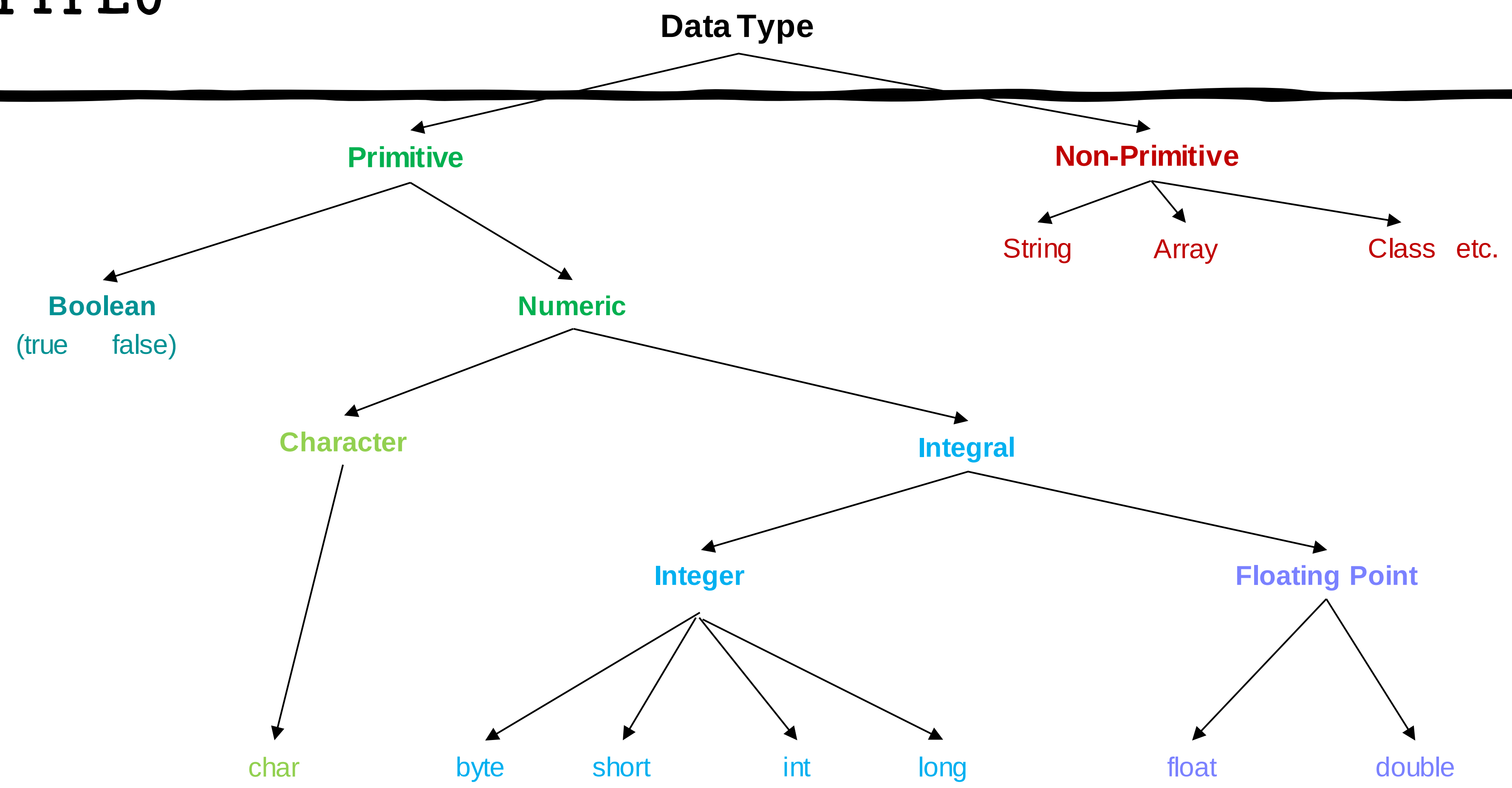
INTRODUCTION

- Programs are built to manipulate information - in a manner that is
  - logical
  - efficient
  - Effective
  - Scalable
  - Non-redundant

- Many higher-level programs are strongly typed language - i.e.it requires you to be explicit about what kind of information you intend to manipulate
  - e.g. James vs. Jomes, web vs. Web, Pat vs. pat

- And higher-level programs, like Java, supports 2 different kinds of data
  1. Primitive data
  2. Objects

DATA TYPES



Data Type
Primitive
Non-Primitive
String
Array
Class etc.
Boolean
(true false)
Numeric
Character
Integral
Integer
Floating Point
char
byte
short
int
long
float
double

PRIMITIVE TYPES

8 primitive data types

Yes integers are a subset of real numbers - but they are fundamentally different types of numbers, e.g. depending on the type of number we expect we ask:

"how many siblings?" or "how much siblings?"

"how many does it weigh" or "how much does it weigh?"

In programming the distinction is more important because integers and real numbers are represented in a different way in the computer's memory

EXAMPLES OF PRIMITIVE TYPES: FOR NUMBERS, TEXT, ETC.

Type	Description	Examples
int	Integers	2 -56, 0, 56000, -2490
double	real numbers	88.45. -56.0 2490.1234 5.
char	single characters	'c'. 'B' '!' '\n' '?'
boolean	logical values	true. false

EXPRESSIONS

Expressions: set of operations that produce a value e.g.

- (2 * 8) + (5 +6) -1

Operators and operands

Many things to do with expressions

- One of the simplest:
  - System.out.println(42);
  - what is the output?
  - System.out.println(2+8);
  - what is the output?

Try it!

LITERALS

- Literals: the simplest form of expression
- Literals of the type *int*
  - 0, -2349, +567, 4, 93
- Literals of the type *double*.
  - -2349.0 -.82 0.982 64.2 23.
- Literals of type *double* can also be expressed in scientific notation:
  - 2.3e6 1e-5 4.523e34
- Literals of type *char* (character) are enclosed in single quotation marks and can include just one character:
  - 'f' 'C' '!' '3' '\\' '\" '
- Literals of type boolean scores logical information. The 2 keywords that are literal values of type boolean are:
  - true or false

DIVISION & MOD

System.*out*.println(2000004%10);

System.*out*.println(34561236%10);

System.*out*.println(45%5);

System.*out*.println(444%2);

Let's Do this Together!

System.*out*.println(445%2);

System.*out*.println(0%6);

System.*out*.println(6%0);

Q: Why do we care for mod values? What information can mod values give us?

MORE EXAMPLES

- System.***out***.println(42);
- System.***out***.println(2 + 8);
- System.***out***.println("2 + 8");
- System.***out***.println(19 % 5);
- System.***out***.println(2 + 5);
- System.***out***.println(2.0 + 5);
- System.***out***.println(2.5 + 5);

What is the output?
What do you notice?

EVEN MORE EXAMPLES (FOR PRACTICE)

```
System.out.println(19 / 5 + "\n");
System.out.println(207 / 10);
System.out.println(1 / 2);
System.out.println(2 / 8);
System.out.println(2. / 8);
System.out.println(19. / 6);
System.out.println(19 / 6.0);
System.out.println(19 / 6.0000);
System.out.println(19.0 / 6);
System.out.println(19.00 / 6);
```

Let's Do This Together.
Then run the code line by line

Real number example

```
2.0 * 2.4 + 2.25 * 4.0 / 2.0
   \_____/
      |
     4.8    + 2.25 * 4.0 / 2.0
                 \_____/
                    |
     4.8    +      9.0    / 2.0
                     \________/
                         |
     4.8    +           4.5
        \_________________/
                 |
                9.3
```

PRECEDENCE
(TRY THIS & CHECK ON ECLIPSE)

```
System.out.println(3 * 9 + -2 + 10 / 5 - (10 % 2))
```

- Ans:

What is the output?

```
System.out.println(((3 * 9 + -2 + 10 / 5 - (10 % 2)) + (3 * 9 + -2 + 10 / 5 - (10 % 2))) * (10 % 2)
```

- Ans:

Within the same level of precedence, the operators are evaluated in one direction - usually left to right

Description	Operators
unary operators	+, - e.g. -2, 7
multiplicative operators	*, /, %
additive operators	+, -

MIXING TYPES & CASTING

When `int` and `double` are mixed, the result is a `double`.

`4.2 * 3` is `12.6`

2.0 + 10 / 3 * 2.5 - 6 / 4

Try it!

The conversion is per-operator, affecting only its operands.

```
7 / 3 * 1.2 + 3 / 2
 \_/
  |
  2   * 1.2 + 3 / 2
  \_____/
     |
    2.4     + 3 / 2
              \___/
                |
```

Try it!

```
What did you expect?
What is the answer

Eclipse E
```

MIXING TYPES & CASTING

- Suppose you have some books that are 0.15 ft wide and you want to know how many will fit in a shelf that is 2.5 ft wide
  - Which of the below calculations would be appropriate to use?
  - System.*out.println*(2.5 / .15);
  - System.*out.println*(2 / .15);
  - System.*out.println*((int) 2.5 / .15);
  - System.*out.println*((int) (2.5 / .15));

Groups

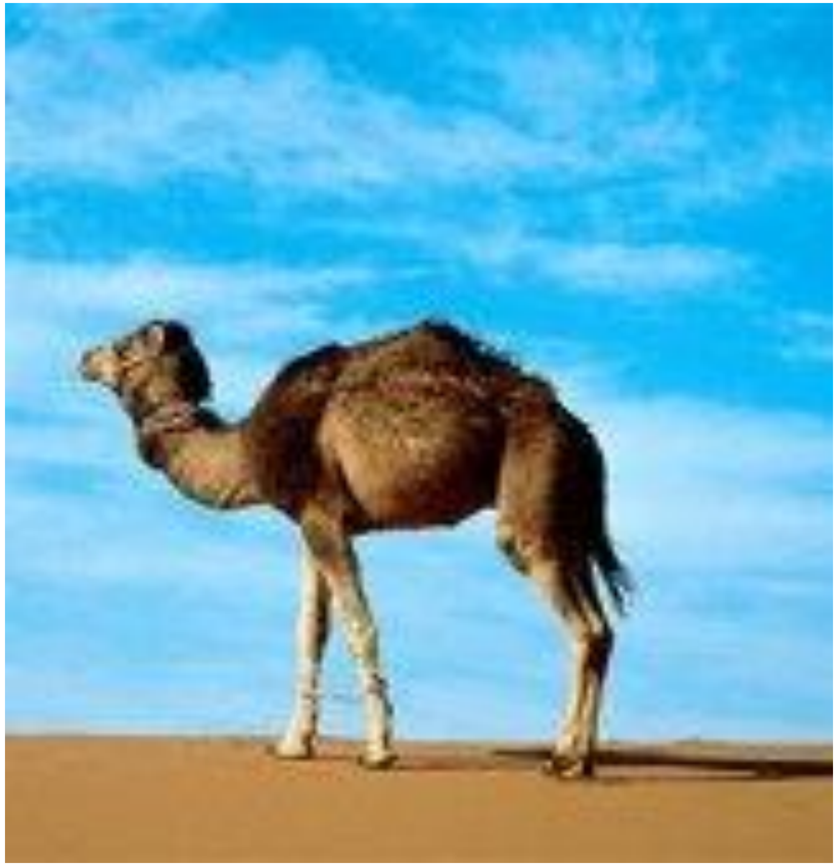
VARIABLES

- Primitive data can be stored in the computer's memory as a variable
  - *type*
  - *name*
  - *this will store a value*

- Imagine variables being placed in cells - and Java is very picky as to what kind of data must be placed in those cells e.g. if you tell Java you want to store a variable of *type* int, then you have to be sure to do so. Likewise with char and double etc.

- You have to decide on what you want to *name* your variable

- Variable names can be camel case for easier reading e.g. camelCase



DECLARING VARIABLES

- Declaration: a request to set aside a new variable with a given *type* and *name*
- <type> <name>; e.g. double height;
  - Unassigned variable
- Once a variable has been declared, the computer will set aside a memory location to store its value

WHAT IS WRONG THIS CODE?

```
public class Receipt {
    public static void main(String[] args) {
        // Calculate total owed, assuming 8% tax / 15% tip
        System.out.println("Subtotal:");
        System.out.println(38 + 40 + 30);
        System.out.println("Tax:");
        System.out.println((38 + 40 + 30) * .08);
        System.out.println("Tip:");
        System.out.println((38 + 40 + 30) * .15);
        System.out.println("Total:");
        System.out.println(38 + 40 + 30 +
                           (38 + 40 + 30) * .08 +
                           (38 + 40 + 30) * .15);
    }
}
```

We'll fix this together soon!

ASSIGNING VALUES

```
public class E {

   public static void main(String[] args) {
       int x = 1;
       int y = x+1;
       int z = x+y;

  System.out.println(x + y + z);

}
```

Try it together
Run the code

ASSIGN VALUES (STUDENT WORK)

```
public class StuffIntDoub {
        public static void main(String[] args) {
                int x = 1;
                int y = 2;
                double z = 3.0;
                double m = 3.9;

                double q = x + y + z;
                System.out.println(q);

                int p = (int)( x + y + z);
               //int p = ( x + y + z);
                double r = x + y +(int) m;
                System.out.println(p + ", " + r);
        }
}
```

STRING CONCATENATION

```
public class StuffConcat {
    public static void main(String[] args) {
        System.out.println(1+2+3);

        System.out.println("1"+"2"+"3");

        System.out.println("hello" +1 + 2 + 3);

        System.out.println(1+2+"hello" +3+4);


        System.out.println(”bye"+9*3);

        System.out.println(”bye"+9+3*12);

        System.out.println(9+3*12+”bye"+9+3*12);


        System.out.println("1"+1);

        System.out.println(4-1+"abc");

        System.out.println("abc"+4-1); // your thoughts and solution?
    }
}
```

Try it together
Run the code
Line by line

USEFULNESS OF STRING CONCATENATION

```
public class G {

    public static void main(String[] args) {
        int x = 1;
        int y = x+1;
        int z = x+y;
     System.out.println("x, y, z respectively are: "+x + ","+y +","+ z);

     x = 6;
     y = x*z;
     z = y*y;
    System.out.println("x, y, z respectively are: "+x + ","+y +","+ z);
    }
}
```

USEFULNESS TO STRING CONCATENATION

```
public class StuffConcat2 {

    public static void main(String[] args) {

        double grade = (95.1 + 71.9 + 82.6) / 3.0;
        System.out.println("Your grade is " + grade);

        int students = 11 + 17 + 4 + 19 + 14;
        System.out.println("There are " + students + " students in the course.");

    }

}
```

Try it!

FIX "BAD" RECEIPT CODE (GROUP WORK)

- Use variables (How many variables?
- Name variables appropriately (e.g. subtotal, tax etc.)
- Use concatenation

- Sample output for "good" receipt program

```
Subtotal: 108.0
Tax: 8.64
Tip: 16.2
Total: 132.84
```

CHECKING IN

- How are you keeping up with CSCI 161?
- Do you look over the material after every new lecture?
- Do you try out the code?
- Are you getting to know your classmates (through informal groups AND through class group work)?
- Autolab and grades
- All due dates are listed on D2Lm
- Class attendance

CHANGE- USING ONLY DIVISION (/) & MOD (%): GROUP WORK

- Consider the following:
  - You have 92 cents, how many quarters can you obtain?
  - What is left over? What is a simple calculation for that?
  - Now, how many dimes can you obtain?
  - What is left over? What is a simple calculation for that?
  - Now how many nickels can you obtain?

DO THE PRE-LAB EXERCISE ON D2L

YOU ARE READY FOR LAB 3!

- Let's Do It!

FYI: THE 8 PRIMITIVE TYPES

Type	Description/ Values	Size	Range
byte	twos complement integer/ signed integers	8 bits	-128 to 127
short	twos complement integer/ signed integers	16 bits	-32768 to 32767
int	twos complement integer/ signed integers	32 bits	-2,147,483,648 ..2,147,483,647
long	twos complement integer/ signed integers	64 bits	-9,223,372,036,854,775,808 .. 9,223,372,036,854,775,807
float	IEEE 754 floating point	32 bits	-3.4E+38 to +3.4E+38
double	IEEE 754 floating point	64 bits	-1.7E+308 to +1.7E+308
char	Unicode character	16 bits	https://unicode-table.com/en/
boolean	true, false	1 bit used in 32-bitinteger	N/A