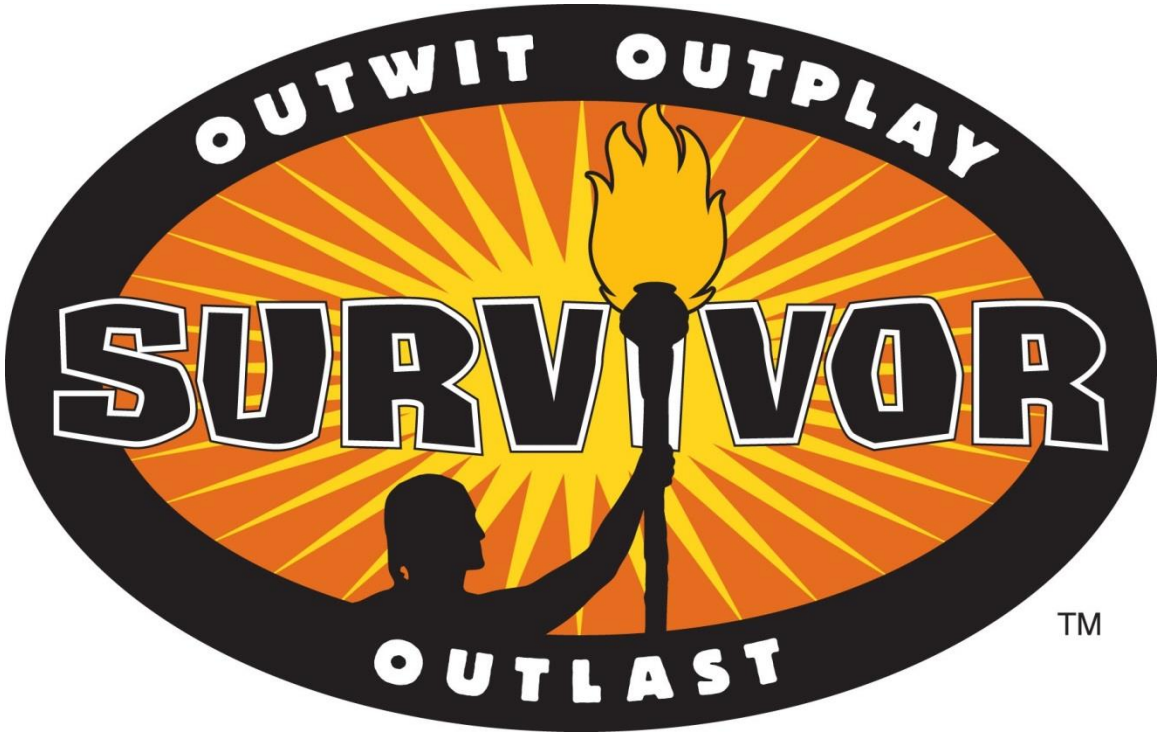
Survivor: CSCI 135



Variables and data types

- Variables
  - Stores information your program needs
  - Each has a unique name
  - Each has a specific type

Java built-in type	what it stores	example values	operations
`String`	sequence of characters	`"Hello world!"` `"I love this!"`	concatenate
`char`	characters	`'a', 'b', '!'`	compare
`int`	integer values	`42` `1234`	add, subtract, multiply, divide, remainder
`double`	floating-point values	`9.95` `3.0e8`	add, subtract, multiply, divide
`boolean`	truth values	`true` `false`	and, or, not

Some definitions

```
int a;
```

Declaration statement

"I'm going to need an integer and let's call it a"

NOTE: in Java you are *required* to declare a variable before using it!

```
a = 10;
```

Variable name

"Whenever I say a, I mean the value stored in a"

Literal

"I want the value 10"

= in CS is not the same as = in math!

```
int b;
b = 7;
```

Assignment statement

"Variable b gets the literal value 7"

```
int c = a + b;
```

Combined declaration and assignment

"Make me an integer variable called c and assign it the value obtained by adding together a and b"

Text

- `String` data type
  - A sequence of characters
  - Double quote around the characters
  - Concatenation using the + operator

```
String firstName = "Keith";
String lastName  = "Vertanen";
String fullName  = firstName + " " + lastName;
String favNumber = "42";

System.out.println(fullName +
                   "'s favorite number is " +
                   favNumber);
```

```
Keith Vertanen's favorite number is 42
```

Characters

- char data type
  - Holds a single character
  - Single apostrophe, e.g. 'a', 'z'

```
public class CharExample
{
   public static void main(String [] args)
   {
      char ch1 = 'y';
      char ch2 = 'o';
      String result = "" + ch1;

      result = result + ch2;
      result = result + ch2;
      result = result + ch2;

      System.out.println(result);
   }
}
```

Double quotes with nothing in between, an empty String

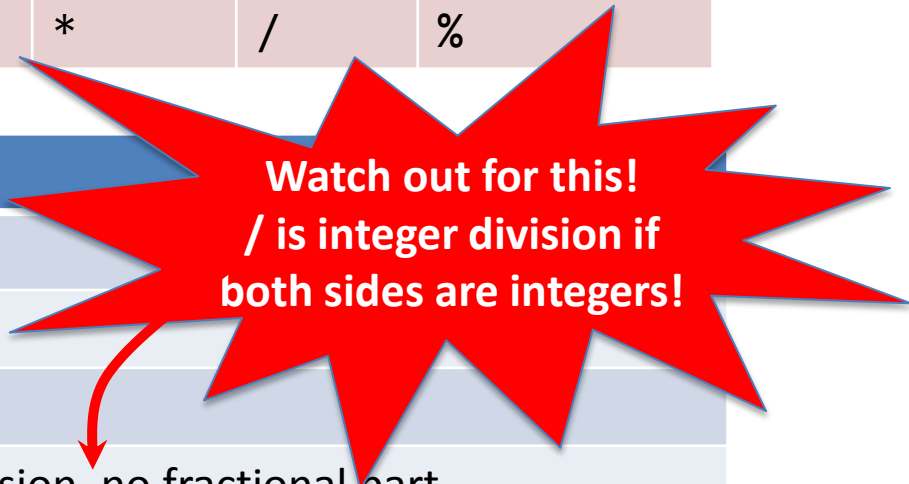
```
% java CharExample
yooo
```

Integers

- `int` data type
  - An integer value between -2^{31} and $+2^{31}-1$
    - Between -2,147,483,648 and 2,147,483,647
  - Operations:

add	subtract	multiply	divide	remainder
+	-	*	/	%

example	result	comment
`10 + 7`	17	
`10 - 7`	3	
`10 * 7`	70	
`10 / 7`	1	integer division, no fractional part
`10 % 7`	3	remainder after dividing by 7
`10 / 0`		runtime error, you can't divide an integer by 0!



Integers

- `int` data type
  - Normal rules of mathematical precedence
    - e.g. multiplication/division before addition/subtraction
  - Use ()'s to force a different order of calculation

example	result	comment
`10 + 7 * 2`	24	multiplication comes before addition
`(10 + 7) * 2`	34	()'s force addition to occur first
`10 / 7 + 2`	3	integer division result is 1 which is added to 2
`10 - 7 - 2`	1	
`(10 - 7) - 2`	1	
`10 - (7 - 2)`	5	

Floating-point numbers

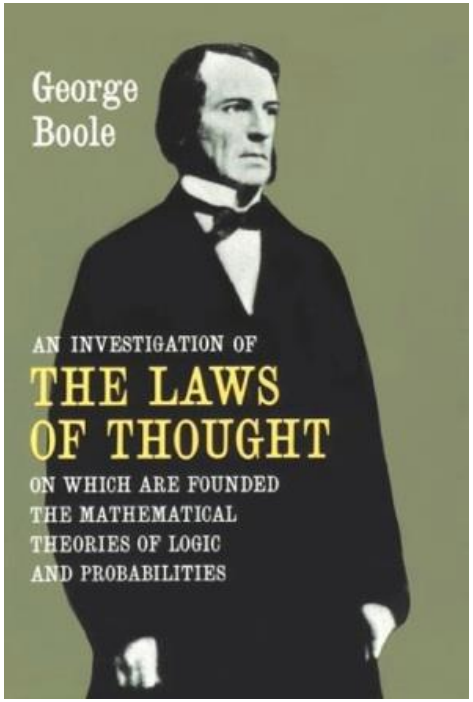
- `double` data type
  - Floating-point number (as specified by IEEE 754)
  - Operations:

add	subtract	multiply	divide
+	-	*	/

example	result
`9.95 + 2.99`	`12.94`
`1.0 - 2.0`	`-1.0`
`1.0 / 2.0`	`0.5`
`1.0 / 3.0`	`0.3333333333333333`
`1.0 / 0.0`	`Infinity`
`0.0 / 123.45`	`0.0`
`0.0 / 0.0`	`NaN`

Booleans

- `boolean` data type
  - Either `true` or `false`
  - Controls logic and flow of control in programs
  - Operations:



logical AND	logical OR	logical NOT
&&	||	!

Note: two symbols for logical AND and OR, not one!

Booleans

- `boolean` data type

logical AND	logical OR	logical NOT
&&	||	!

`!a` → "Is a set to `false`?"
`a && b` → "Are both a *and* b set to `true`?"
`a || b` → "Is either a *or* b (or both) set to `true`?"

a	!a
true	**false**
false	**true**

a	b	a && b	a || b
false	false	**false**	**false**
false	true	**false**	**true**
true	false	**false**	**true**
true	true	**true**	**true**

Comparisons

- Given two numbers → return a `boolean`

operator	meaning	true example	false example
==	equal	`7 == 7`	`7 == 8`
!=	not equal	`7 != 8`	`7 != 7`
<	less than	`7 < 8`	`8 < 7`
<=	less than or equal	`7 <= 7`	`8 <= 7`
>	greater than	`8 > 7`	`7 > 8`
>=	greater than or equal	`8 >= 2`	`8 >= 10`

Is the sum of `a`, `b` and `c` equal to `0`?
```
(a + b + c) == 0
```

Is `grade` in the `B` range?
```
(grade >= 80.0) && (grade < 90.0)
```

Is `sumItems` an even number?
```
(sumItems % 2) == 0
```

Type conversion

- Java is strongly typed
  - Helps protect you from mistakes (aka "bugs")

```
public class TypeExample0
{
   public static void main(String [] args)
   {
      int orderTotal = 0;
      double costItem = 29.95;

      orderTotal = costItem * 1.06;
      System.out.println("total=" + orderTotal);
   }
}
```

```
% javac TypeExample0.java
TypeExample0.java:7: possible loss of precision
found   : double
required: int
      orderTotal = costItem * 1.06;
                           ^
```

Type conversion

- Converting from one type to another:
  - Manually → using a cast
    - A cast is accomplished by putting a type inside ()'s
  - Casting to `int` drops fractional part
    - Does not round!

```
public class TypeExample1
{
   public static void main(String [] args)
   {
      int orderTotal = 0;
      double costItem = 29.95;

      orderTotal = (int) (costItem * 1.06);

      System.out.println("total=" + orderTotal);
   }
}
```

```
% java TypeExample1
total=31
```

Type conversion

- Automatic conversion
  - Numeric types:
    - If no loss of precision → automatic promotion

```
public class TypeExample2
{
   public static void main(String [] args)
   {
      double orderTotal = 0.0;
      int costItem = 30;

      orderTotal = costItem * 1.06;

      System.out.println("total=" + orderTotal);
   }
}
```

```
% java TypeExample2
total=31.8
```

Type conversion

- Automatic conversion
  - `String` concatenation using the + operator converts numeric types to also be a `String`

```
public class TypeExample3
{
   public static void main(String [] args)
   {
      double costItem = 29.95;

      String message = "The widget costs ";
      message = message + costItem;
      message = message + "!";

      System.out.println(message);
   }
}
```

```
% java TypeExample3
The widget costs 29.95!
```

Converting text to a numeric type

method	description
`Integer.parseInt(String a)`	converts text a into an `int`
`Double.parseDouble(String a)`	convert text a into a `double`

```
public class CostCalc
{
   public static void main(String [] args)
   {
      String product = args[0];
      int    qty     = Integer.parseInt(args[1]);
      double cost    = Double.parseDouble(args[2]);

      double total = qty * cost;

      System.out.print("To buy " + qty);
      System.out.print(" " + product);
      System.out.println(" you will need $" + total);
   }
}
```

```
% java CostCalc elections 2 1e6
To buy 2 elections you will need $2000000.0
```

Control flow

- Interesting and powerful programs need:
  - To skip over some lines
  - To repeat lines
- Conditionals → sometimes skip parts
- Loops → allow repetition of lines

if statement

- Most common branching statement
  - Evaluate a `boolean` expression, inside the ()'s
  - If `true`, do some stuff
  - [optional] If `false`, do some other stuff

```
if (expression)
{
    statement1;
    statement2;
    …
}
```

Curly braces used to denote a code "block":
All lines in block get executed (in sequence) or none of the them do

Note lack of semicolon!

```
if (expression)
{
    statement1;
    statement2;
    …
}
else
{
    statement3;
    statement4;
    …
}
```

if statement

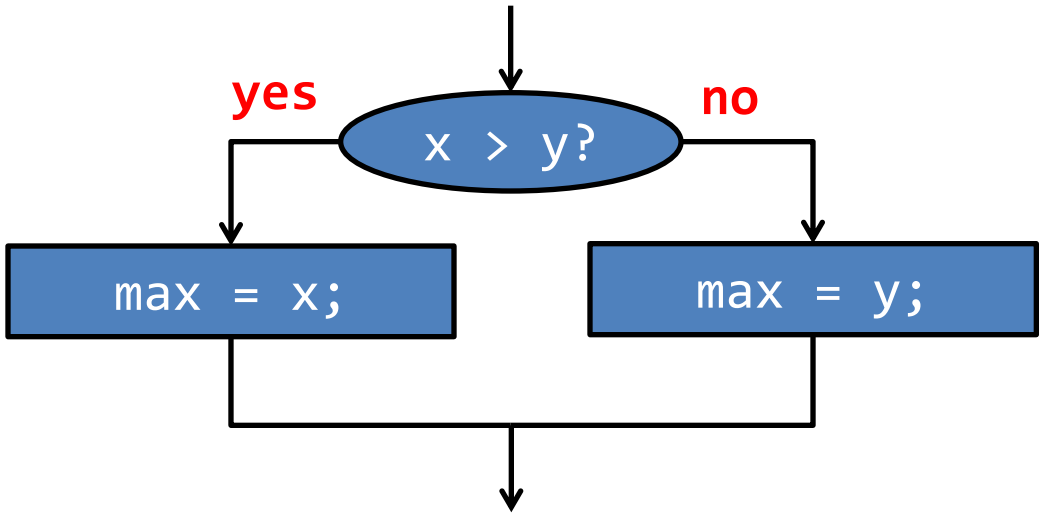
- {}'s optional if only one statement

```
if (expression)
   statement1;
```

```
if (expression)
   statement1;
else
   statement2;
```

- Example:

```
if (x > y)
   max = x;
else
   max = y;
```



if examples

```
if (x < 0)
   x = -x;
```

Take absolute value of x

```
if (Math.random() < 0.5)
   System.out.println("heads");
else
   System.out.println("tails");
```

Flip a fair coin and print out the results.

```
if (x > y)
{
   int t = x;
   x = y;
   y = t;
}
```

Put x and y into sorted order

```
num = 0;
if (args.length > 0)
{
   num = Integer.parseInt(args[0]);
}
```

If a command line option is passed in, use it as the value for num.

Nested if

- Execute one of three options:

```
if (category == 0)
{
   title = "Books";
}
else
{
   if (category == 1)
   {
      title = "CDs";
   }
   else
   {
      title = "Misc";
   }
}
```

==

```
if (category == 0)
{
   title = "Books";
}
else if (category == 1)
{
   title = "CDs";
}
else
{
   title = "Misc";
}
```

  - Both do exactly same thing
  - Right probably more readable in general

Data Types & Conditionals

Write a Java program to convert a temperature in Fahrenheit to a temperature in kelvin or vice versa. The conversion equation is:

$$T_k = [\frac{5}{9}T_f - 32.0] + 273.15$$

The user will input the temperature and its units on the command line and you will convert it to the other unit. For example, if the user types:

java <temppgm> 32 F

Your program should convert it to kelvin, and if the user types:

java <temppgm> 32 K

Your program should convert it to Fahrenheit.

VERY IMPORTANT: Name your program <yourusername>1.java
For example, my program would be named mvandyne1.java

Data Types & Conditionals

The cost of sending a package by an express delivery service is $15.00 for the first two pounds, and $5.00 for each pound or fraction thereof over two pounds. If the package weighs more than 70 pounds, a $15.00 excess weight surcharge is added to the cost. No package over 100 pounds will be accepted. Write a Java program that accepts the weight of a package in pounds on the command line and computes the cost of sending the package. Be sure to handle the case of overweight packages.

For example, if the user types:

java <weightpgm> 55

Your program should compute the cost of mailing a package weighing 55 pounds.

VERY IMPORTANT: Name your program <yourusername>2.java
For example, my program would be named mvandyne2.java