Université Concordia University
CENTRE FOR CONTINUING EDUCATION

INTRODUCTION TO PROGRAMMING WITH JAVA - CEJV416

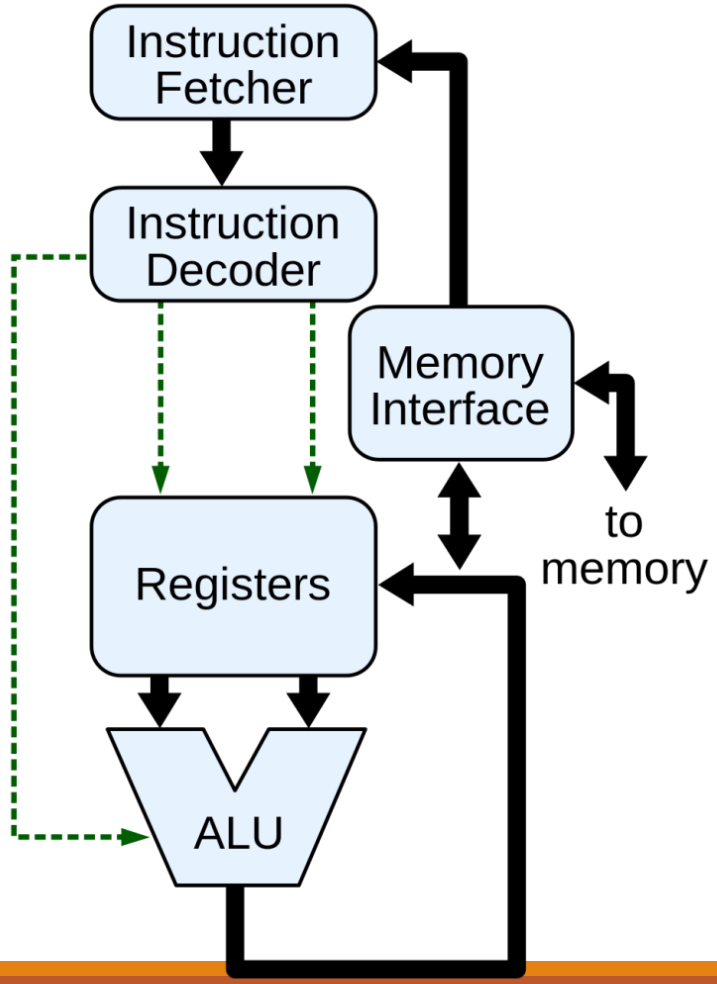
LECTURE #2

Data Types
Variables
Arithmetic
Strings

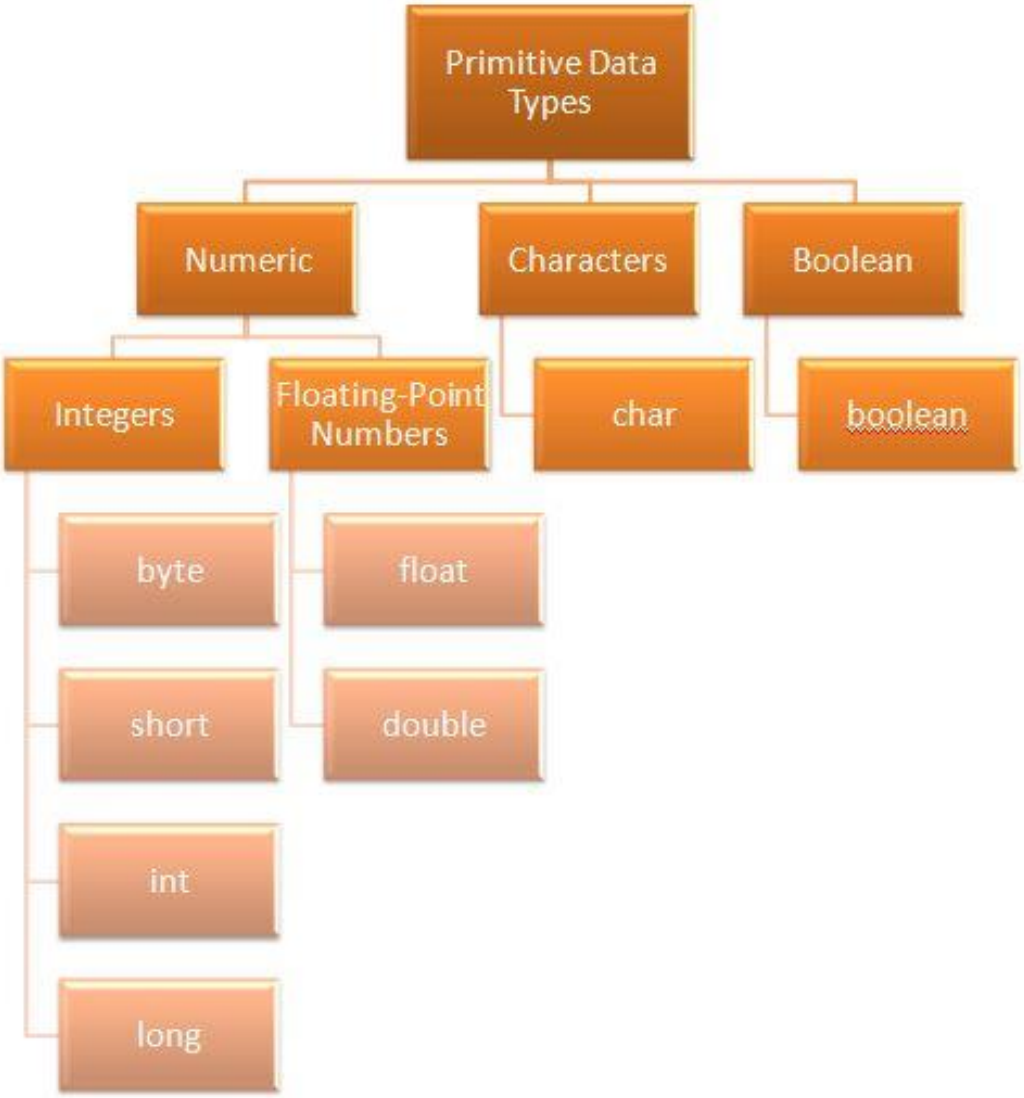
What does the CPU do?

- Fetches data from RAM
- Fetches instructions from RAM
- Instructions can be
  - Load and store
  - Branching
  - Math
  - Logic/Comparisons
- Carries out the instructions sequentially
- The instructions, the data, and the results are kept in memory in the CPU called registers
- Results are copied from the registers to RAM

CPU Block Diagram



Primitive Data Types



Describing the world with numbers

Numbers are used to describe or measure just about everything in the world

- How many chickens on the farm?
- How much gas does a car consume?
- What is the size of a file on disk?
- How many kilometers to the grocery store?
- What is the temperature?

Describing the world with words

Numbers are difficult to remember

Words are easier to remember

We assign names to numbers

- What colour is the paint?
- What is the product code?
- What is the address?
- What model of car do you drive?

Integers

Whole numbers

- Without a fractional part

Positive or negative

Ordinal

- Representing a position
- First, second, third

Cardinal

- Representing a count
- 100 employees
- 12 bagels

Integer Types

Four different integer types

All are signed (negative or positive)

Distinguished by the number of bytes of RAM they use

`byte`

- 1 byte
- -128 to 127

`short`

- 2 bytes
- -32,768 to 32,767

Integer Types

`int`

- 4 bytes
- -2,147,483,648 to 2,147,483,647

`long`

- 8 bytes
- -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

`byte` and **`short`** are used in embedded systems when RAM and disk space is small

The best integer type is

- ***int*** is the most common integer type
- Use ***long*** if you really need very large integers

Numbers inside the CPU

- CPUs describes the world around it with just two possible numbers:
  - 0 and 1
- Only two states in a computer, circuit opened and circuit closed
  - Electricity or no electricity
- Base 2 or Binary Number System
- Every integer decimal (base 10) number can be mapped to a binary (base 2) number

Imprecision – Floating point numbers

Numbers with a fractional part

- May be precise such as 3.5
- May be infinite such as 3 1/3
- May be irrational like π (pi)

Infinite and irrational numbers cannot be mapped precisely to binary

Imprecision – Floating point numbers

- Two floating point data types

  `float`
  - 4 bytes
  - -3.4E38 to 3.4E38 with up to 7 significant digits

  `double`
  - 8 bytes
  - -1.7E308 to 1.7E308 with up to 15 significant digits

- Just say **no** to **`float`**
- Only ever use **`double`**
- Never compare floating point for equality
- There are special classes available when precision is required

A single character

Characters such as the letters of the alphabet are represented by the char

`char`

- 2 bytes
- Represents the Unicode character set
- Range of 0 - 65,535
- First 128 characters map to ASCII
  - A is 65
  - B is 97
  - space is 32

True or False

`boolean`

- Size varies
- Range of values is **`true`** to **`false`**

Used in logical operations that form the basis of decision making in code

Some languages support 0 as false and not 0 as true (such as 1) so that integers can be used for logical expressions

Java does not support this and only the **`boolean`** data type can be used.

The Variable

- All data is stored in the computer's memory at a unique address
- Remember that we have trouble remembering numbers such as addresses
- We are unconcerned with the exact address that our data is at
- As long as we can access it we are happy
- Variable is a name that refers to a specific data type at a specific location in memory
- We use the variable name and the compiler handles the translation to the address when the program executes

The identifier

Any time we name something such as a variable we create an identifier

Identifiers are arbitrary names

Should describe what they represent

Basic rules:

- Start with a letter, underscore, or dollar sign
- Use letters, dollar signs, underscores, or digits for subsequent characters
- Use up to 255 characters
- Don’t use Java keywords

Identifier conventions

- Conventions are guidelines that programmers follow to enhance readability of code
- For variables the convention is called mixed case
- First character should be a letter in lower case
- Switch to uppercase to represent a new word
- Underscores can be used to indicate new words in the identifier but should not be the first character
- Numbers are acceptable
- Do not use the dollar sign

Which of the followings are acceptable variable identifiers?

1. a
2. $currentBalance
3. numberOfStudents
4. average Age
5. the_width_of_the_page
6. 2ndName
7. Username
8. the2ndName
9. long

1. Not good
2. Not good
3. Great!
4. Not acceptable
5. Great!
6. Not acceptable
7. Not good
8. Great!
9. Not acceptable

Which of the followings are acceptable variable identifiers?

1. annualSalary
2. 8cars
3. kilometerPerHour
4. amin@home
5. shoe_size
6. num-oranges
7. Rate2013
8. Volatile
9. balesOfHay

1. Great!
2. Not acceptable
3. Great!
4. Not acceptable
5. Great!
6. Not acceptable
7. Great!
8. Not acceptable
9. Great!

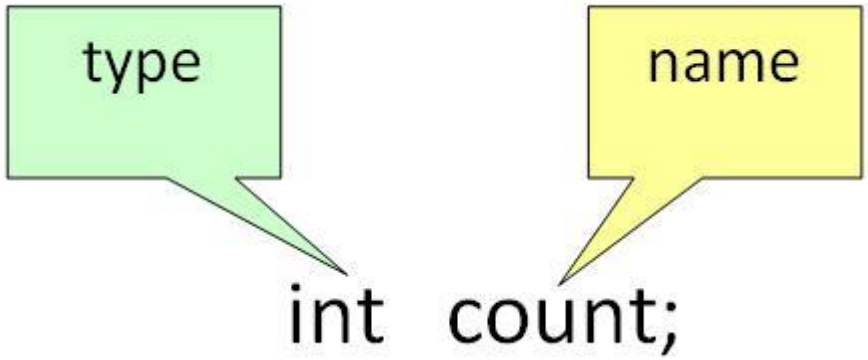
Identifiers for variables should be nouns

Keywords

abstract	continue	for	new	switch
assert	default	if	package	synchronized
boolean	do	goto	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Declaring variables

To represent data in a program you must first declare a variable




```
double annualSalary;
int balesOfHay;
```

```
char choice;
boolean success;
```

Initializing a variable

Initial value can be assigned in the declaration

```
int count = 100;
double annualSalary = 29344.87;
char choice = 'A';
int balesOfHay = 215;
boolean success = false;
```

Assigning a new value to a variable

- Once declared a variable can have a new value assigned to it

```
count = 27;

annualSalary = 43125.90;

choice = 'D';

balesOfHay = 14;

success = true;
```

- No limit to the number of times you can assign to a variable
- Most important rule is that the value on the right hand side must be the same type as the variable on the left hand side
- Variables with a value can be used on the right hand side

```
count = balesOfHay;
```

final Variables

- Variables can be declared that cannot be assigned a value more than once Called constants
- Declaration begins with the keyword final
- Can be initialized or assigned later

```
final int DAYS_IN_NOVEMBER = 30;

final double GST;

GST = 0.05;
```

- Conventions says that identifiers for ***constants are in uppercase only***
- Attempts to assign a new value generates an error

How to code assignment statements

```
int quantity = 0;                       // initialize an
                                        // integer variable
int maxQuantity = 100;                  // initialize another
                                        // integer variable

// two assignment statements
quantity = 10;                          // quantity is now 10
quantity = maxQuantity;                 // quantity is now 100
```

The basic operators for arithmetic expressions

Operator	Name
`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division

Statements that use simple arithmetic expressions

```
// integer arithmetic
int x = 14;
int y = 8;
int result1 = x + y;          // result1 = 22
int result2 = x - y;          // result2 = 6
int result3 = x * y;          // result3 = 112
int result4 = x / y;          // result4 = 1

// double arithmetic
double a = 8.5;
double b = 3.4;
double result5 = a + b;       // result5 = 11.9
double result6 = a - b;       // result6 = 5.1
double result7= a * b;        // result7 = 28.9
double result8 = a / b;       // result8 = 2.5
```

Exercise

Write a program that will print the result of following operations:

```
// integer arithmetic
int x = 14;
int y = 8;
int result1 = x + y;          // result1 = 22
int result2 = x - y;          // result2 = 6
int result3 = x * y;          // result3 = 112
int result4 = x / y;          // result4 = 1

// double arithmetic
double a = 8.5;
double b = 3.4;
double result5 = a + b;       // result5 = 11.9
double result6 = a - b;       // result6 = 5.1
double result7= a * b;        // result7 = 28.9
double result8 = a / b;       // result8 = 2.5
```

Statements that increment a counter variable

```
int invoiceCount = 0;
invoiceCount = invoiceCount + 1;       // invoiceCount = 1
invoiceCount = invoiceCount + 1;       // invoiceCount = 2
```

Statements that add amounts to a total

```
double invoiceAmount1 = 150.25;
double invoiceAmount2 = 100.75;
double invoiceTotal = 0.0;
invoiceTotal = invoiceTotal + invoiceAmount1;
                                       // invoiceTotal = 150.25
invoiceTotal = invoiceTotal + invoiceAmount2;
                                       // invoiceTotal = 251.00
```

Statements that mix int and double variables

```
int result9 = invoiceTotal / invoiceCount
                                       // result9 = 125
double result10 = invoiceTotal / invoiceCount
                                       // result10 = 125.50
```

Arithmetic Operations

Operator	Name	Example	Result
+	Addition	int a = 2 + 2;	4
-	Subtraction	int a = 4 - 2;	2
*	Multiplication	int a = 2 * 3;	6
/	Division	float f = 3 / 2; int a = 3 / 2;	1.5 1
%	Modulus	int a = 8 % 5;	3
++	Increment	int a = 2; a++;	3
--	Decrement	int a = 2 a--;	1

The String

Strings consist of 0 or more characters.

Can represent any sequence of characters

- Moose
- 42366X788
- 5551212
- The entire contents of a novel

Dynamic

- No fixed size
- Is as large as required

The String

Not a primitive

String is a Java class

The only class that supports the assignment operator =

```
String city = "Montreal";
city = "Toronto";
```

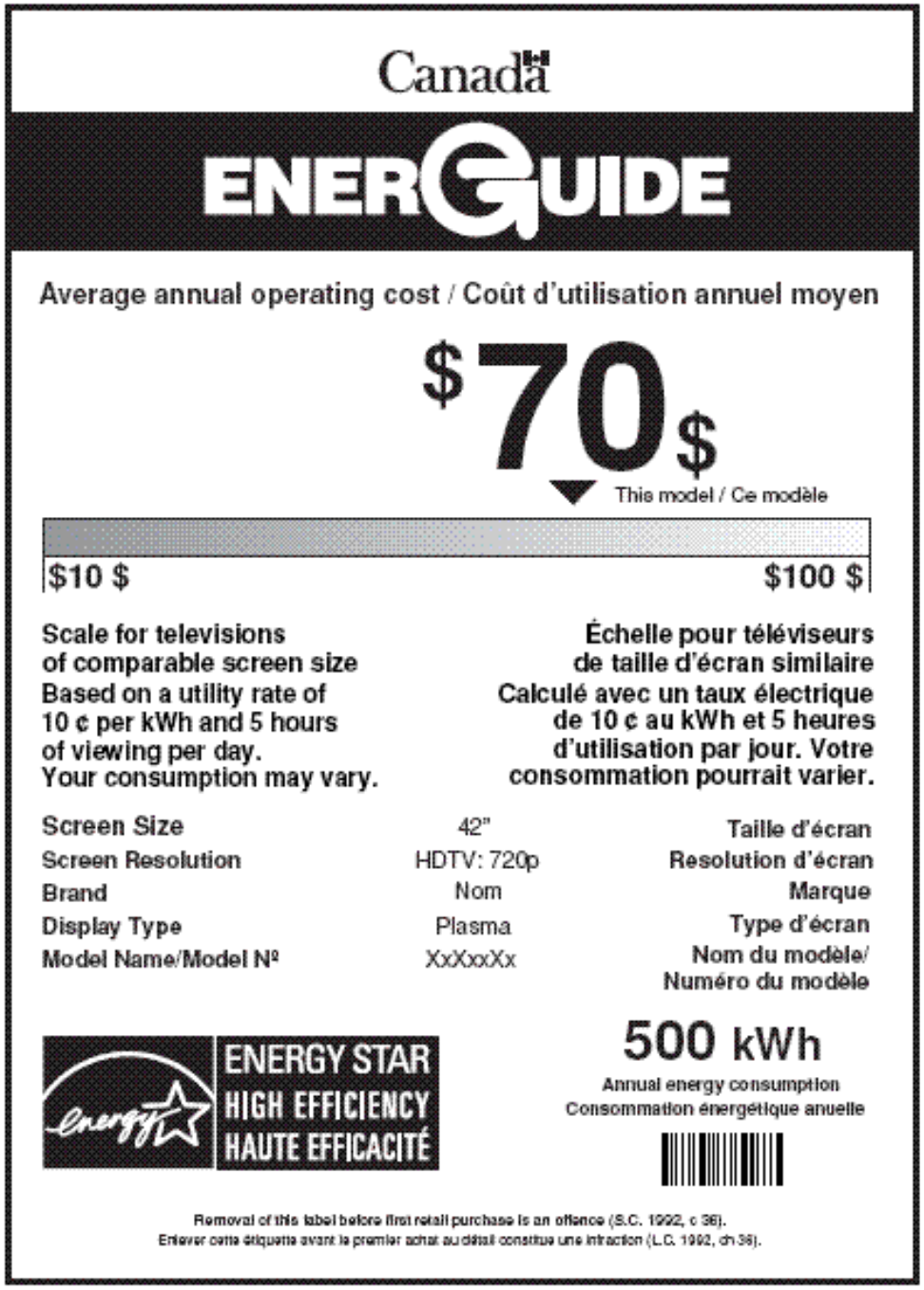
If it isn't a number, true or false, or a single character* then its likely a String

Strings can be empty

```
String city = "";
```

Strings can be null

```
String city;
```


Canada
ENERGUIDE
Average annual operating cost / Coût d'utilisation annuel moyen
70
This model / Ce modèle
$10 $
$100 $
Scale for televisions of comparable screen size Based on a utility rate of 10 ¢ per kWh and 5 hours of viewing per day. Your consumption may vary.
Échelle pour téléviseurs de taille d'écran similaire Calculé avec un taux électrique de 10 ¢ au kWh et 5 heures d'utilisation par jour. Votre consommation pourrait varier.
Screen Size 42" Taille d'écran
Screen Resolution HDTV: 720p Resolution d'écran
Brand Nom Marque
Display Type Plasma Type d'écran
Model Name/Model Nº XxXxxXx Nom du modèle/ Numéro du modèle
energy
ENERGY STAR
HIGH EFFICIENCY
HAUTE EFFICACITÉ
500 kWh
Annual energy consumption
Consommation énergétique anuelle
Removal of this label before first retail purchase is an offence (S.C. 1992, c 36).
Enlever cette étiquette avant la premier achat au détail constitue une infraction (L.C. 1992, ch 36).