Basics of Java

Lecture 3

Java vs other programming languages

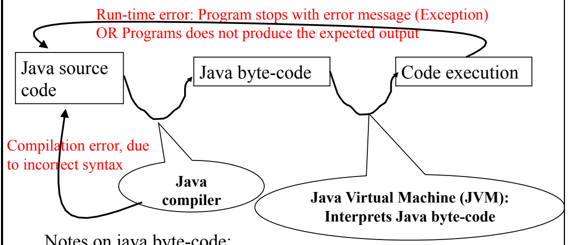
- Basic syntax very similar to C, C++
- Fully object-oriented: all code and data is within a class
- Java handles memory management: no need to allocate or free memory!
- No pointers, no segmentation faults!!

$\Downarrow$

- Easy to learn and use
- Execution time slightly slower than C or C++

Programming cycle in Java

[Downey Ch 1]


Run-time error: Program stops with error message (Exception)
OR Programs does not produce the expected output
Java source code
Java byte-code
Code execution
Compilation error, due to incorrect syntax
Java compiler
Java Virtual Machine (JVM): Interprets Java byte-code

Notes on java byte-code:

Advantages: - Byte-code is platform independent
- Very important for internet!

Disadvantage: - Byte-code has to be interpreted by the JVM so it runs slightly slower

Integrated Development Environment

- IDE: Program that facilitates writing code, compiling it, running it, and debugging it.
- Recommended IDE: Eclipse
  - Freely available at
    http://www.eclipse.org/downloads/index.php
  - You also need to install a Java Runtime Environment (JRE), from the same URL.
  - Runs on all OS: Windows, Mac, Linux, etc.
  - Installed on all machines in Trottier

My first Java program

[Downey Ch 2]

```
/* This programs prints a welcoming statement */
public class Welcome {
   // Every executable class have to contain a method called main like below
   // When the class is executed, main is the first method to be called
  public static void main(String args[]) {
      System.out.println("Welcome to Java!");
      System.out.println("This is easy!");
  }
}
```

File: Welcome.java

1) Compile Welcome.java
2) Run the program

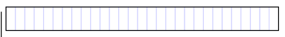
```
Output:      Welcome to Java!
             This is easy!
```

Variables and types

[Downey Ch 2]

- Variable: temporary storage location in memory. It has
  - a name (to refer to it)
  - a type (to describe what kind of information it can store).
  - a value (content stored in memory)
- Two kinds of types:
  - Primitive types (seen today)
  - Classes (next lecture)

Memory (RAM)



```
public class VariablesExample {
public static void main(String args[]) {
    int age;      // age can store an integer
    float pi;     // float can store a decimal number
     age = 29;
     pi = 3.14;
    }
}
```

[Downey Ch 2]

Primitive types

Type	Size	Description	Range
byte	8-bit	signed integer	[-128,127]
char	16-bit	integer	[0, 65536] (encodes 'a', 'b' …)
short	16-bit	signed integer	[-32768,32767]
int	32-bit	signed integer	[-2147483648, 2147383647]
long	64-bit	signed integer	[-9223372036854775807, 9223372036854775806]
float	32-bit	decimal number	1.40239e-45 to 3.402823e+38
double	64-bit	decimal number	4.9406e-324 to 1.79769e+308
boolean	8-bit	boolean	true or false

[Downey Ch 2]

Expressions and Assignments

- **Expression:** Piece of code that has a value of a certain type
- **Assignment:** Storing the value of an expression into a variable
- **Syntax:** <variable> = <expression>

Examples:

Memory (RAM)

```
public class Expressions {
public sta[illegible]main(String args[]) {
    int i, j[illegible]
    float f, [illegible]
    i = 5;
    j = i + 1;
    k = k * 2;  // Compiling error: Why? _______________________________
    j = j / 2;
    j + 10;      // Legal, but useless. Why? _______________________________
    g = j + 3.14; // Note the implicit conversion of j into a float
    f = 15 / 2;    // f now has value 7.0. Why?________________________________
    f = ((float) i) / 2 // Explicit type conversion (casting) of i into a float. f is now 7.5
}}
```

[Downey Ch 6.5-6.6]

Boolean expressions

- Boolean expressions have value true or false.
- Operations on booleans:
  - NOT : !
  - AND : &&
  - OR : ||

```
boolean a,b;
int i = 5;
into j = 2;
a = ( i * j < 10 );             // a is ______
b = !a;                         // b is ______
a = (i+j < 10) && (i+j > 0);  // a is ____
b = ( i<0 ) || ( j>1 ) ;          // b is _____
a = ! ( b && ( i>0 || j<i ) ); // a is ______
```

Exercise

public class Exercise {	i	j	f	a	b	c
public static void main(String args[]) {						
int i,j;	-	-				
float f;	-	-	-			
boolean a, b;	-	-	-	-	-	
char c = 'f' ;	-	-	-	-	-	'f'
f = i; // compilation error: i is not initialized						
i = 9;	9	-	-	-	-	'f'
a = (f > 100); // compilation error: f is not initialized						
f = i;	9	-	9.0	-	-	'f'
b = true;	9	-	9.0	-	T	'f'
a = (b || (12345.67*i - f/0.02345 == 0.003464));	9	-	9.0	T	T	'f'
j = i;	9	9	9.0	T	T	'f'
j = j + 1; // value of j: 10, value of i is still 9	9	10	9.0	T	T	'f'
i = f + 3.3; // error: a float value cannot be stored in an int						
i = (int) (f + 3.3); // the float value 12.3 is cast into an int. // It becomes 12, so i becomes 12	12	10	9.0	T	T	'f'
b = b && ((i == j) || (!b || f > 10));	12	10	9.0	T	F	'f'
}}						

[Downey Ch 4.1-4.4]

Conditionals

- Syntax: if (<boolean expression>) <statementBlock1>
  [else <statementBlock2>]
- Executes <statementBlock1> only if <boolean expression> is true. Otherwise <statementBlock2> is executed.

```
/* Determines if a point (x,y) is inside a circle of radius r centered at (a,b) */
if ( (a-x)*(a-x) + (b-y)*(b-y) <= r*r ) {
    System.out.println("The point is inside the circle");
    if ( (x==a) && (y==b) ) System.out.println("It is the center");
}
else {
    System.out.println("The point is outside the circle");
    // other statements could be here
}
```

- Note: If the statement block contains a single statement, then the {} can be omitted.

[Downey Ch 7.1-7.3]

while loops

- Syntax:
  while (<boolean expression>) <statementBlock>
- Keeps executing <statementBlock> repeatedly as long as <boolean expression> is true.
  If <boolean expression> is false from the beginning, then <statement> is never executed.

```
int n = 32;
int i = 0;
int exp = 1;
while (exp < n) {
   i++;          // equivalent to i = i +1;
   exp = exp * 2;  // we could also write exp *= 2;
}
// What is the value of exp and i at the end? ________________________
```

[Downey Ch 7.1-7.3]

do-while loops

- do <statementBlock> while (<boolean expression>)
- Same as while-loop but <boolean condition> is checked *after* executing <statementBlock>, so <statement> is always executed at least once.

```
// Keep asking for a price as long as the number entered is not positive
double price = 0;
String line;     // String is a special type of variable. More about strings next week
do {
    System.out.println("Enter price of item:");
    line = stndin.readLine();                    // Read a line from keyboard
    price = Double.parseDouble(line) ;     // Parse the line to get a double
} while (price<=0);
```

[Downey Ch 12.4]

For loops

- for (<statement1>; <boolean expression>; <statement2>) <statementBlock3>
- Equivalent to:

```
<statement1>
while (<boolean expression>) {
   <statementBlock3>
   <statement2>
}
```

What does this print?

____

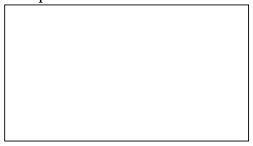
```
int n=5;
int s=0;
int i;
for (i = 0; i < n; i++ ) {
    s = s + i;
}
System.out.println("Value of s:" + s);
```

What does this print?

```
int n=5;
for(int i = 1; i <= n; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print( i );
    }
    System.out.println(""); // print the end of line
}
```

Memory (RAM)

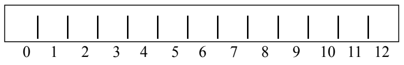
Output



[Downey Ch 12]

Arrays

- Arrays are used to store and manipulate several variables of the same type
- Array X:



- To access the i-th element:

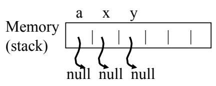
```
i = 5;
X[i] = 2;
int a = X[i] + 1;
```

- Note: first element is at index 0.
- X.length is the number of elements in X (here, it's 13)
- Java makes sure you don't write outside arrays:
  - ArrayIndexOutOfBoundsException gets thrown if you try

[Downey Ch 12]

Arrays in Java

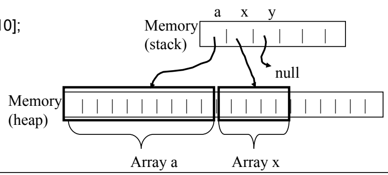
- Declaration:



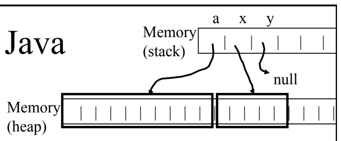
```
double[] a;   // a will be array of double.
int[] x,y;    // x and y will both be arrays of ints
// IMPORTANT: At this point, a, x, and y are references to null arrays
a[ 3 ] = 2;  // would cause error because a is null
```

- Allocation

```
a = new double[10];
x = new int[5];
```



Arrays in Java

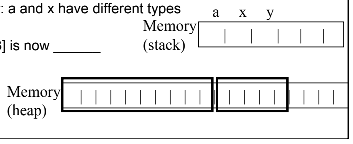


- Accessing elements

```
a[ 3 ] = 3.1;
x[ 0 ] = 1;
a[ x[ 0 ] ] = 1.5;
y[ 3 ] = 2;    // compiling error. Why? ___________________
x[ a[ 1 ] ] = 3; // compiling error. Why? __________________________
a[ 10 ] = 3; // run-time error. ArrayIndexOutOfBoundsException gets thrown
```

- Assignments with arrays

```
y=x;
y[ 3 ]=9;      // the value of x[3] is now _______
x[ 2 ]=4;      // the value of y[2] is now _______
a = x;          // compil. error: a and x have different types
x = new int[ y[2] ];
x[3] = 5;    // the value of y[3] is now ______
y = new int[3];
```



[Downey Ch 12]

Multi-dimensional arrays

- Arrays can have more than one dimension:

```
double matrix [ ] [ ] = new double[ 10 ] [ 10 ];
// initialize the matrix to zero
for ( int i=0; i<10; i++ ) {
   for ( int j=0; j<10; j++ ) {
      matrix[ i ][ j ] = 0;
    }
}

// make it an identity matrix
for ( int i=0; i<10; i++ ) matrix[ i ][ i ]=1;
```