[JAVA TRAINING]

Providing Basic Knowledge of Java

By TrungHuynh
Email: trunghuynh.cse@gmail.com

Outline

- **Basic Java Programming**
  - Hello world program
  - Variables
  - OOP
- **Java Conventions**
- **Java Performance**
- **Mini Project**

HELLO WORLD

- **Creating a file helloworld.java**
- **Trying to compile it and launch it from command line**

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

Variable

- **In Java, every variable has to stay in a class**
- **There are three kinds of variables:**
  - Instance variable – None static variable
  - Class variable – Static variable
  - Local variable
  - (Argument variable)

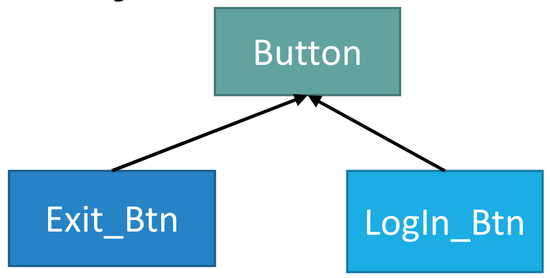
```
public class CoreMWEcimSWMTest {
		public static final String resourcePath= “/home/cluster/incoming”;
		public String UpgradeName = “ECIMTEST1”;


		public clusterReboot(){
				int retry = 5;
				String result = null;
				while( retry > 0){
						result = sshlib.sendCommand(“cluster reboot –a –q”);
						assertEqual(“1”, result);
		}
}

CoreMWEcimSWMTest test1;
test1.clusterReboot();
System.out.println(“resourcePath = “ + CoreMWEcimSWMTest.resourcePath);
```

OOP – Object Oriented Programming

Access Levels table in Java				
Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

OOP – Object Oriented Programming

- **Parent class object will be created before child class object**
- **But the child object will be destroyed before its parent object**



- **Java does not support multiple inheritances but a class can implement for multiple interfaces**

  public class Engineer implements Employee, Person {}

  ~~public class Engineer extends Employee, Person {}~~

NOTES

- **Memory allocation of object will be collected by Grabage Colletor of JVM if there is no any reference to it -> assign Null after we don’t use object anymore.**
- **Array(static array in c/c++) vs ArrayList , Vector(dynamic array in c/c++)**

OutOfMemoryError

- If your Java program is running out of memory, there are several things you can do.
    1. Make unused objects garbage collectable.
    2. Avoid excessive object creation.
    3. Allocate more memory for the heap.
    4. Choose an alternative technique (eg, caching).

OOP – Object Oriented Programming

- **Polymorphism**
  - Up-casting
  - Down-casting

→

Used to parameter input
Used to a member of class

```
Person p;
Employee e = new Employee();
p = (Person) e;
p.setName(...);
p.setSalary(...); // compile error
```

```
String teamInfo(Person p1, Person p2) {
return "Leader: " + p1.getName() +"; member: " + p2.getName();}...
Employee e1, e2; Manager m1, m2;...
System.out.println(teamInfo(e1, e2));
teamInfo(m1, m2); teamInfo(m1,e2);
```

```
class Manager extends Employee {
Employee assistant;...
public void setAssistant(Employee e) {
                assistant = e;
                }...
}...
Manager junior, senior;...
senior.setAssistant(junior);
```

Java Conventions

- **Interface:**
  - Class names should be nouns, in mixed case with the first letter of each internal word capitalized.
- **Method:**
  - Methods should be verbs
  - In mixed case with the first letter lowercase, with the first letter of each internal word capitalized.
  - Example: run(), runFast(), getBackground().
- **Variable:**
  - Lowercase first letter and internal words start with capital letters
- **Constant:**
- Uppercase with words separated by underscores (“_”).
- Example: int MIN_WIDTH = 4; int MAX_WIDTH = 999; int GET_THE_CPU = 1;

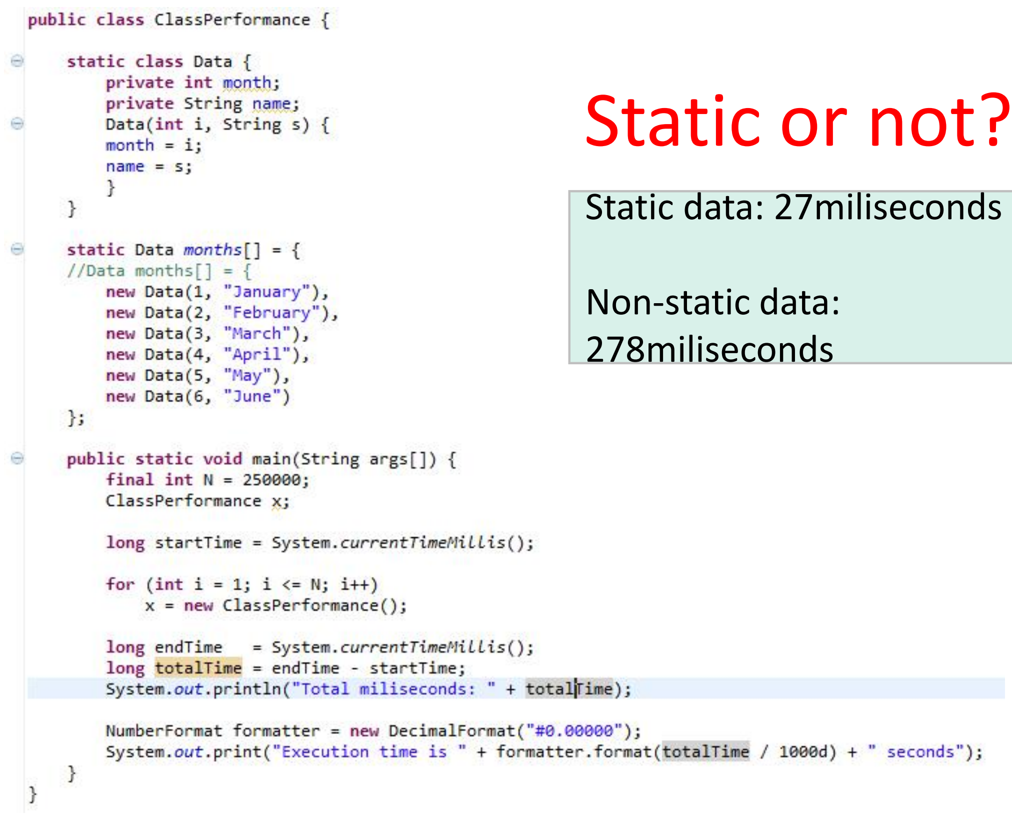
Java Conventions

- **Indentation:**
  - Suggesting use only the tab (8/4 space tab)
- **Wrapping lines:**
  - Break before an operator
- **{}**

```
try {
        //
        //
} catch ( Exception e){
}
```

Improve Performance - Class

- **Instance Initialization**



```
public class ClassPerformance {

    static class Data {
        private int month;
        private String name;
        Data(int i, String s) {
        month = i;
        name = s;
        }
    }

    static Data months[] = {
    //Data months[] = {
        new Data(1, "January"),
        new Data(2, "February"),
        new Data(3, "March"),
        new Data(4, "April"),
        new Data(5, "May"),
        new Data(6, "June")
    };

    public static void main(String args[]) {
        final int N = 250000;
        ClassPerformance x;

        long startTime = System.currentTimeMillis();

        for (int i = 1; i <= N; i++)
            x = new ClassPerformance();

        long endTime   = System.currentTimeMillis();
        long totalTime = endTime - startTime;
        System.out.println("Total miliseconds: " + totalTime);

        NumberFormat formatter = new DecimalFormat("#0.00000");
        System.out.print("Execution time is " + formatter.format(totalTime / 1000d) + " seconds");
    }
}
```

Static or not?

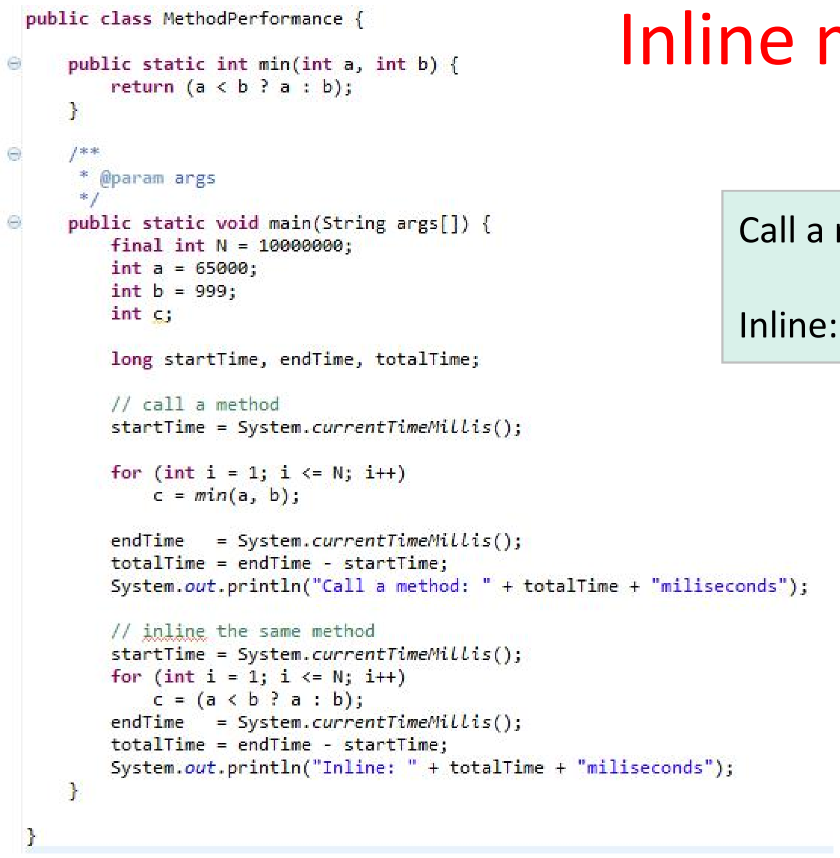
Static data: 27miliseconds

Non-static data: 278miliseconds

Improve Performance – Method (1)

- **Inline Method**

Inline method or not?



```
public class MethodPerformance {

    public static int min(int a, int b) {
        return (a < b ? a : b);
    }

    /**
     * @param args
     */
    public static void main(String args[]) {
        final int N = 10000000;
        int a = 65000;
        int b = 999;
        int c;

        long startTime, endTime, totalTime;

        // call a method
        startTime = System.currentTimeMillis();

        for (int i = 1; i <= N; i++)
            c = min(a, b);

        endTime   = System.currentTimeMillis();
        totalTime = endTime - startTime;
        System.out.println("Call a method: " + totalTime + "miliseconds");

        // inline the same method
        startTime = System.currentTimeMillis();
        for (int i = 1; i <= N; i++)
            c = (a < b ? a : b);
        endTime   = System.currentTimeMillis();
        totalTime = endTime - startTime;
        System.out.println("Inline: " + totalTime + "miliseconds");
    }

}
```

Call a method: 416miliseconds

Inline: 204miliseconds

Improve Performance – String (1)

- Remember: Java strings are Immutable
- It meant, if you do like this:

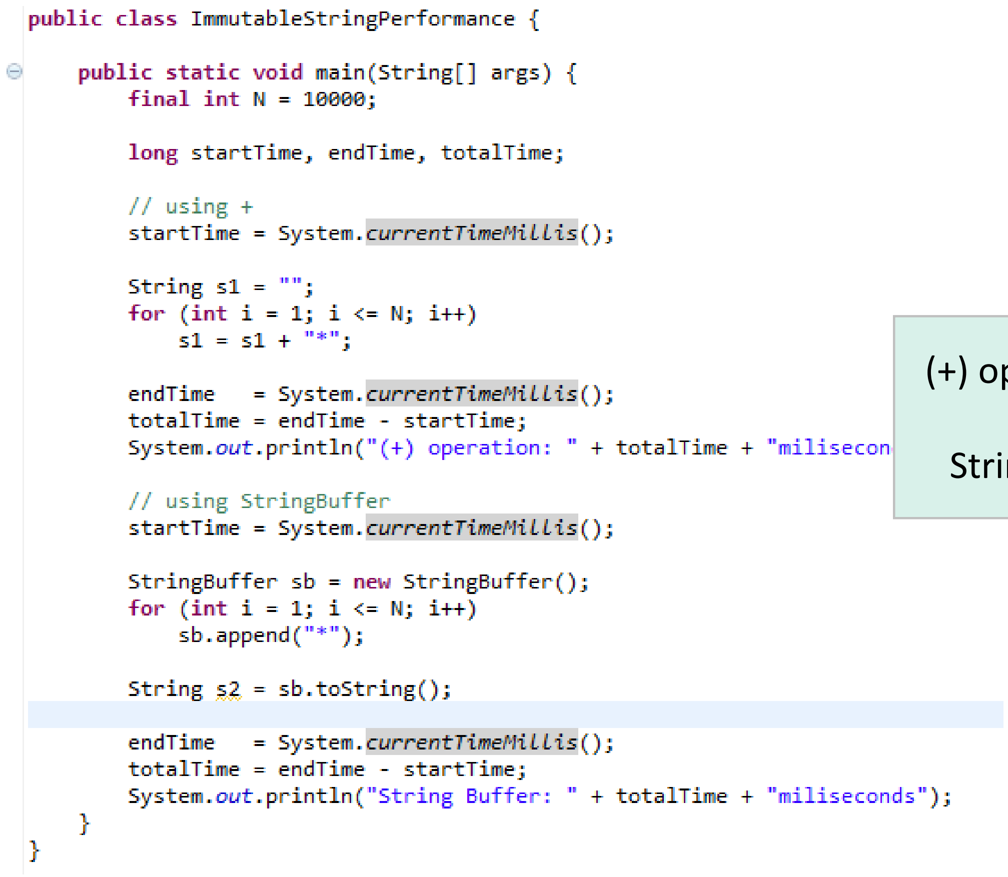
```
String str = “testing”;
str += “abc”;
```

it’ll be translated like this:

```
String str = "testing";
StringBuffer tmp = new StringBuffer(str);
tmp.append("abc");
str = tmp.toString();
```

- The two strings to be concatenated are copied to a temporary string buffer, then copied back.

Improve Performance – String (2)


```
public class ImmutableStringPerformance {

    public static void main(String[] args) {
        final int N = 10000;

        long startTime, endTime, totalTime;

        // using +
        startTime = System.currentTimeMillis();

        String s1 = "";
        for (int i = 1; i <= N; i++)
            s1 = s1 + "*";

        endTime   = System.currentTimeMillis();
        totalTime = endTime - startTime;
        System.out.println("(+) operation: " + totalTime + "milisecon

        // using StringBuffer
        startTime = System.currentTimeMillis();

        StringBuffer sb = new StringBuffer();
        for (int i = 1; i <= N; i++)
            sb.append("*");

        String s2 = sb.toString();

        endTime   = System.currentTimeMillis();
        totalTime = endTime - startTime;
        System.out.println("String Buffer: " + totalTime + "miliseconds");
    }
}
```

(+) operation: 163miliseconds

String Buffer: 4miliseconds

End

- **Thank for your listening**
- **Questions ?**
- **References:**
  - http://introcs.cs.princeton.edu/
  - Think In Java
  - http://docs.oracle.com/javase/tutorial/java/javaOO/classdecl.html