CSE231

Advanced Computer Programming

Course work

- Labs (Theoretical + Practical)
- 2 Quizzes (total 10 , 5 for each) -> time will be discussed
- Final Exam 60
- Midterm Exam 20
- Practical Exam 10

Introduction to Java

- Portable
  - Can be executed on any platform.
- Object oriented
  - Object-oriented programming (OOP) is a popular programming approach that is replacing traditional procedural programming techniques.
- Performance
  - Can be run on any platform without recompiling.

Prerequisites

- Download JDK (J2SE) (**Java Development Kit**)

- software development environment **used** for developing Java applications and applets.
- It includes the Java Runtime Environment (JRE), an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc) and other tools **needed** in Java development.

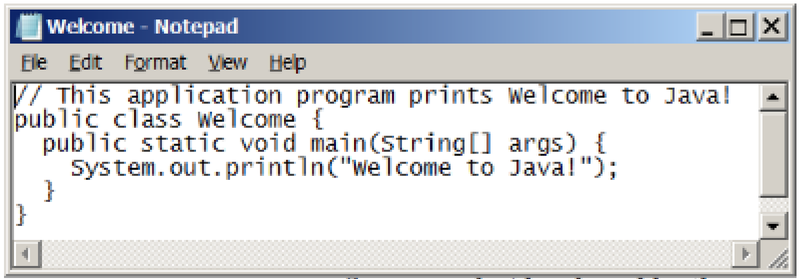
- IDE (Eclipse, Netbeans, IntelliJ IDEA, Oracle Jdeveloper)

Simple Program Java

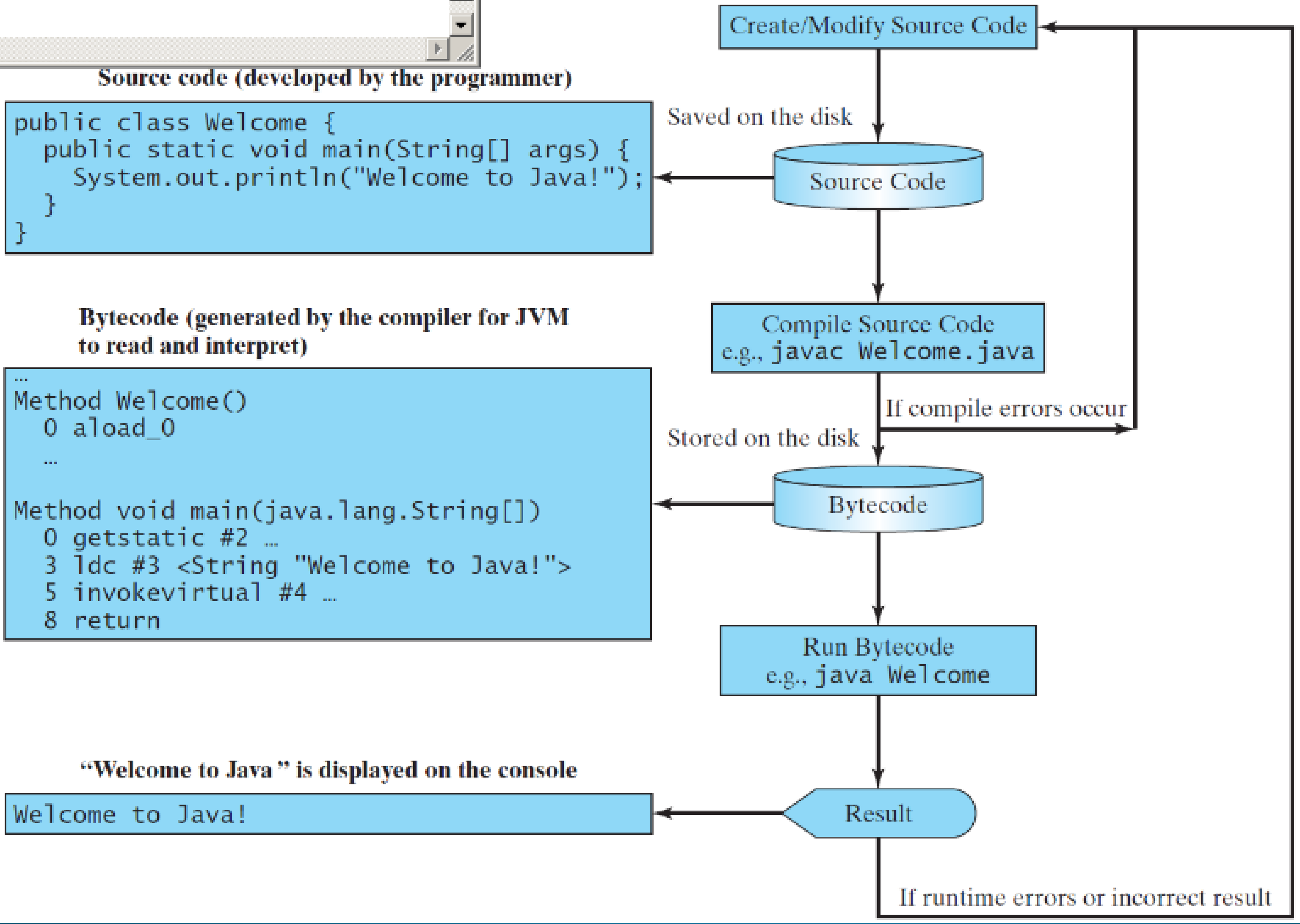
- Anatomy of Java program

```
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Creating, Compiling, and Running Programs



```
// This application program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```



Source code (developed by the programmer)

```
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Bytecode (generated by the compiler for JVM to read and interpret)

```
…
Method Welcome()
  0 aload_0
  …

Method void main(java.lang.String[])
  0 getstatic #2 …
  3 ldc #3 <String "Welcome to Java!">
  5 invokevirtual #4 …
  8 return
```

"Welcome to Java" is displayed on the console

```
Welcome to Java!
```

Variables

- Variables writing conventions -> camelCase

- e.g : int studentId;

- Constants Capitalized

- e.g : int MAX_DURATION;

Methods

- Method writing conventions -> camelCase
- Difference between instance methods, non-instance methods.

  - instance methods: invoked with object

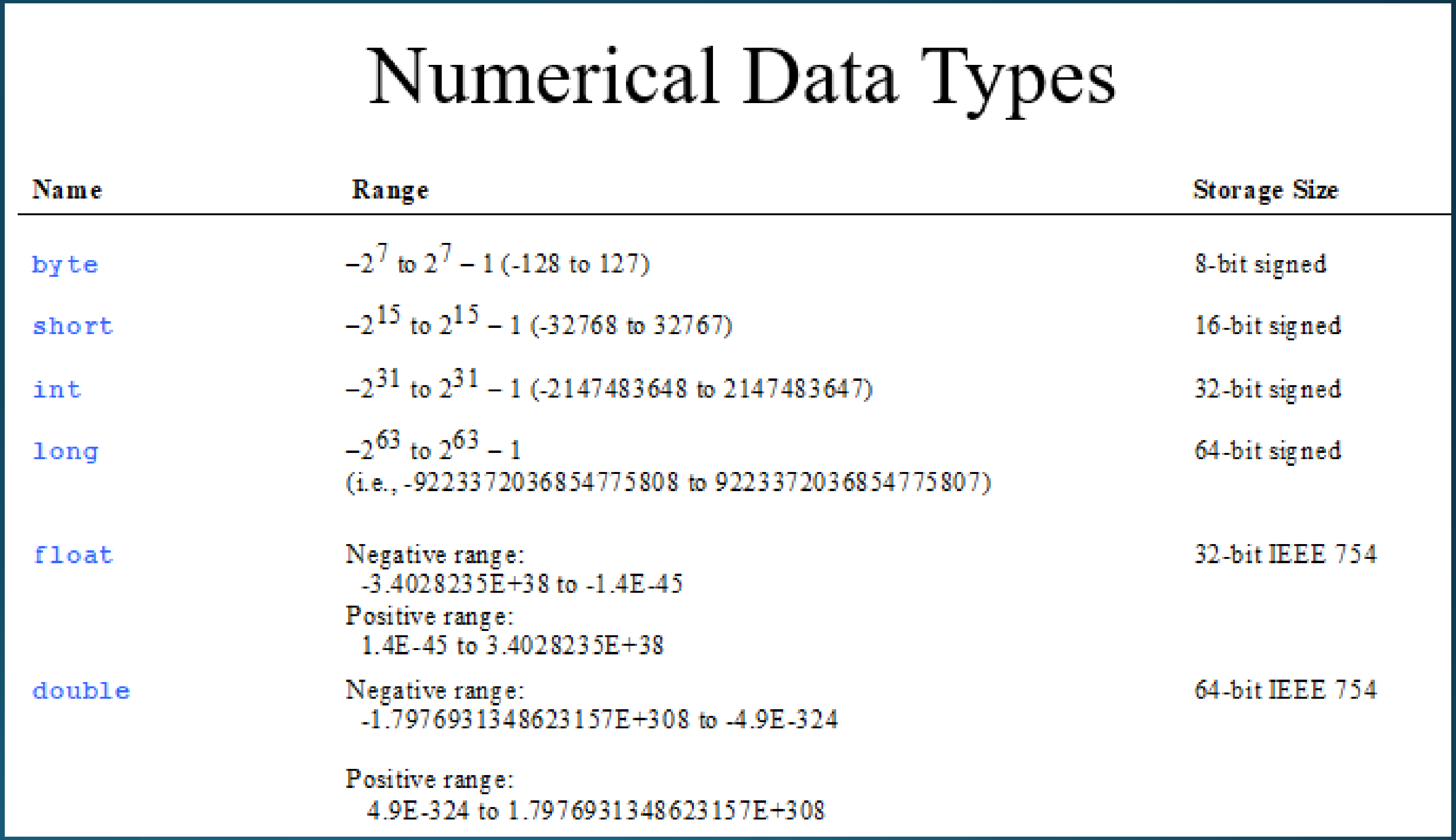
  - non-instance methods (Static methods): invoked with class name

DataTypes

- Other data types as : **Boolean , Boolean, Integer , Double.**

- difference between primitive and Wrapper (object / reference).

- A field, variable or parameter declared as boolean can have values **true** and **false**, while one declared as Boolean can have values **TRUE**, **FALSE** and **null**



Numerical Data Types

Name	Range	Storage Size
`byte`	-2^7 to $2^7 - 1$ (-128 to 127)	8-bit signed
`short`	-2^{15} to $2^{15} - 1$ (-32768 to 32767)	16-bit signed
`int`	-2^{31} to $2^{31} - 1$ (-2147483648 to 2147483647)	32-bit signed
`long`	-2^{63} to $2^{63} - 1$ (i.e., -9223372036854775808 to 9223372036854775807)	64-bit signed
`float`	Negative range: -3.4028235E+38 to -1.4E-45 Positive range: 1.4E-45 to 3.4028235E+38	32-bit IEEE 754
`double`	Negative range: -1.7976931348623157E+308 to -4.9E-324 Positive range: 4.9E-324 to 1.7976931348623157E+308	64-bit IEEE 754

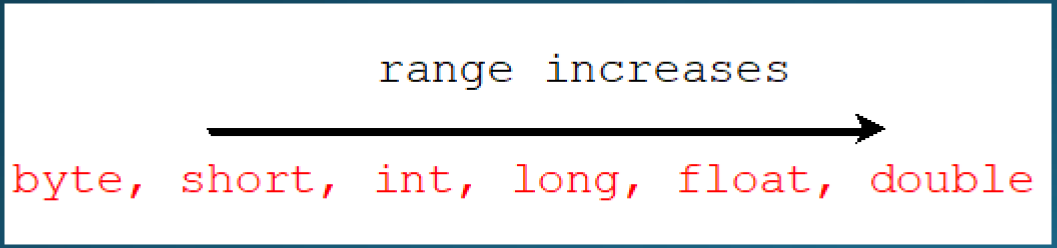
Type Casting

- **Implicit casting**

```
double d = 3;
```

- **Explicit casting**

```
int i = (int)3.0;
int i = (int)3.9;
```

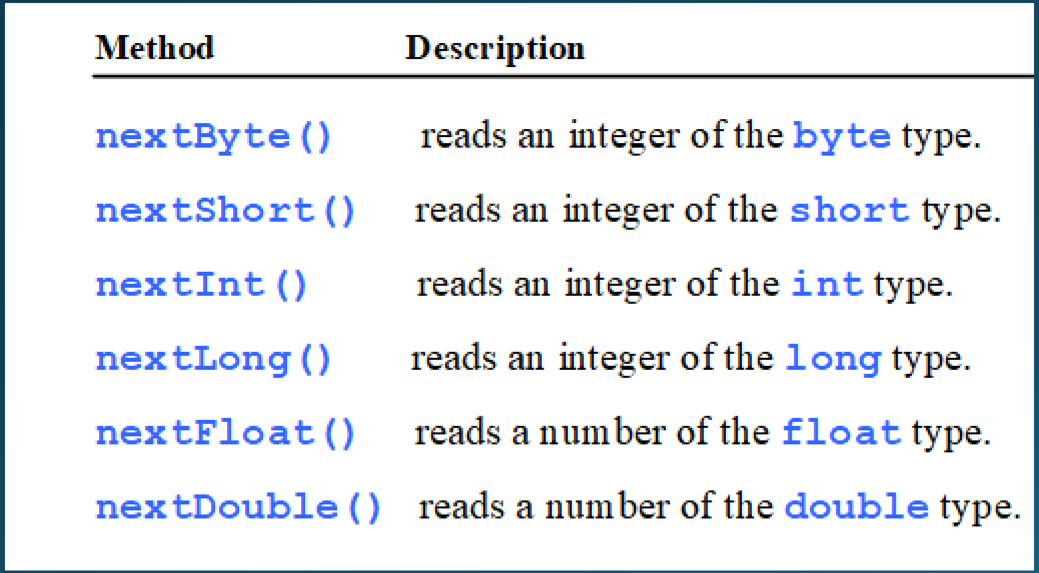


Loops

```
For (int i=0;i<10;i++)
{
// statements
}
```

Reading Input from user

```
Scanner input = new Scanner(System.in);
```



Method	Description
`nextByte()`	reads an integer of the `byte` type.
`nextShort()`	reads an integer of the `short` type.
`nextInt()`	reads an integer of the `int` type.
`nextLong()`	reads an integer of the `long` type.
`nextFloat()`	reads a number of the `float` type.
`nextDouble()`	reads a number of the `double` type.

Math Class

- Contains all trigonometric functions, math operations

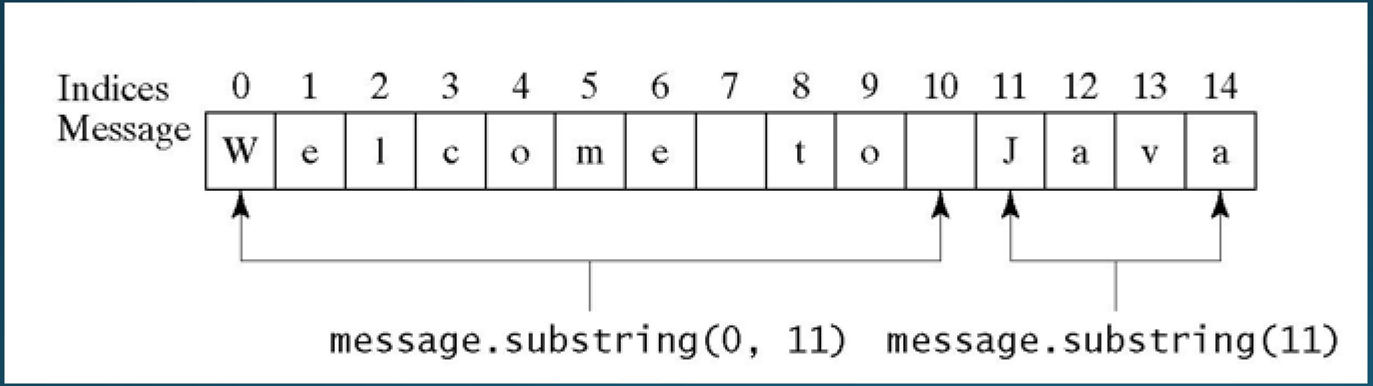
- e.g : exponent , log , power ,...etc.

Arrays

- Data structure to store multiple values of same datatype.
  - `datatype[] arrayRefVar;` **convention**
  - `double arrayRefVar[];` **allowed but not prefered**
- **Operations** : get item at specific index, get length of array
  - `double arrayRefVar[];` **null reference**
  - `double arrayRefVar[] = new double [10];` **initialized**

Strings

- Array of characters
- Operations: getlength () , charAt(int index), concatenation,….



Conditional

```
If (condition)
{
//statements
}
```

- Note :

If with One statement without curly braces is executed based on condition

Switch case

```
switch(expression) {
 case x:
   // statements
   break;
 case y:
   // statements
   break;
 default:
   // statements
}
```

Tips for Lab 1

- Static methods **can** access the **static** variables and **static** methods directly.
- Static methods **can't** access **instance** variables or **instance** methods directly. They must use an object reference to do so.