Object Oriented Programming (CSE 2202)

Chapter 01
Introduction

Objectives

Java Programming

After studying this chapter, students should be able to:

- Learn about Programming Paradigms
  - Structured Vs Object-Oriented paradigm
- Understand OOP Concepts and Features
- Fundamental Java Programming Structures:
  - Tokens: Identifiers, Separators, Operators, keywords & Literals
  - Constants, variables and primitive data types
  - The main() method
  - Control statements: if, switch, while, do…while, for and Jumps in loops
  - Array & Strings in java

Programming Paradigms

Java Programming

- Programming Paradigm is a way of conceptualizing what it means to perform computation and how tasks to be carried out and organized on a computer.

Structured Programming

- Problem solving would involve the analysis of processes in terms of the procedural tasks carried out and the production of a system whose representation is based on the procedural flow of the processes.
- data was separate from code.
- programmer is responsible for organizing everything in to logical units of code/data
- A procedural program is divided into functions, and (ideally, at least) each function has a clearly defined purpose & a clearly defined interface to the other functions in the program.

Disadvantages of Structured Programming

1. **Unrestricted Access**
   - Functions have unrestricted access to global data.
2. **Real-World Modeling**
   - Unrelated functions and data, the basics of the procedural paradigm, provide a poor model of the real world.
3. **Difficult of Creating New Data Types**
   - Traditional languages are not extensible because they will not let you create new data types.

Contd…

OOP Approach

- A modern programming paradigm that allows the Programmer to model a problem in a real-world fashion as an object.
- Major objective is to eliminate some of the flaws encountered in the procedural approach.
- OOP allows us to decompose a problem into number of entities called objects and then build data and methods (functions) around these entities.
- The data of an object can be accessed only by the methods associated with the object
- Follows bottom-up approach in program design

Contd…

Features of OOP

- Emphasis is on data rather than procedure.
- Programs are divided into objects.
- Data Structures are designed such that they characterize the objects.
- Methods that operate on the data of an object are tied together in the data structure.
- Data is hidden and can not be accessed by external functions.
- Objects may communicate with each other through methods.

Basic OOP Concepts

The following are the basic OOP concepts:

1. Objects
2. Classes
3. Data Abstraction
4. Data Encapsulation
5. Inheritance
6. Polymorphism
7. Dynamic Binding
8. Message Passing

Contd…

1. Object

- An object is an any real world entity which may represent place, person, data item related to program.
- An object has **state, behavior and identity**.
- Ex. A 'Mouse' object has,

  State: moving or working

  Behavior: picking or pointing an object on the screen

  Identity: color, company, Identification No. etc.
- An object is a variable/instance of class.
- An object is a run-time entity.

2. Class

- Is the template or blueprint that defines the states and the behaviors common to all objects of a certain kind.
- It is a collection of objects of similar type.
- Classes are user-defined data types & behave like the built-in types of programming language.

Contd…

3. Data Abstraction

- Abstraction means representing essential features without including the background details or explanations
- A classes can use the concept of *Abstraction* and are defined as list of abstract data and functions to operate on these data.
- Since the classes use the concept of Data Abstraction, they are known as *Abstract Data Type(ADT).*

4. Data Encapsulation

- The wrapping up of data and methods into a single unit (called class) is known as encapsulation.
- This insulation of the data from direct access by the program is called data hiding.

Contd…

5. Inheritance

- Is the process by which objects of one class acquire the properties of objects of another class.
- It provides the idea of reusability(reusing the code)

6. Polymorphism

- In polymorphism, '*Poly*' means many and '*morph*' means forms, i.e. *many forms*.
- Is the ability to take more than one form. It allows a function responding in different ways.

7. Dynamic Binding

- Dynamic binding means that the code associated with a given procedure call is not known until the time of the call at runtime.
- Memory is allocated at runtime not at compile time.

8. Message Passing

- The process of invoking an operation of an object is called *Massage Passing*.
- In response to the given massage, the respective method or operation is called.
- OOPs includes objects which communicates by sending/ receiving information with each other.
- Message Passing involves *specifying the name* of the object, *the name of the function* (message) and *the information* to be sent.

```
employee.salary (name);
```

- A message for an object is a request for the execution of a procedure.

Introduction to Java

Java Programming

- Java is an Object Oriented Programming language developed by Sun Microsystems in the year 1991.
- Initially it was named as "Oak" by James Gosling
- In 1995, "Oak" is renamed to "Java" because the name "Oak" did not survive legal registration.
- James Gosling and his team members were consuming a lot of coffee while developing this language.
- Good quality of coffee was supplied from a place called "Java Island'. Hence they fixed the name of the language as Java. The symbol for Java language is *cup and saucer*.
- Sun formally announced Java at Sun World conference in 1995. On January 23rd 1996, JDK1.0 version was released.

Features of Java (Java Buzzwords)

Java Programming

- **Simple:** Learning and practicing Java is easy because of resemblance with C and C++.
- **Object Oriented:** Unlike C++, Java is purely OOP.
- **Distributed:** Java is designed for use on network; it has an extensive library which works in agreement with TCP/IP.
- **Secure:** Java is designed for use on Internet. Java enables the construction of virus-free, tamper free systems.
- **Robust (Strong/ Powerful):** Java programs will not crash because of its exception handling and its memory management features.
- **Portable:** Java does not have implementation dependent aspects and it gives same result on any machine.

Contd…

- **Interpreted:** Java programs are compiled to generate the byte code(.class file). This byte code can be interpreted by the interpreter contained in JVM.
- **Architectural Neutral Language:** Java byte code is not machine dependent, it can run on any machine with any processor and with any OS.
- **High Performance:** Along with interpreter there will be JIT (Just In Time) compiler which enhances the speed of execution.
- **Multithreaded:** Executing different parts of a program simultaneously is called multithreading. This is an essential feature to design server side programs.
- **Dynamic:** We can develop programs in Java which dynamically change on Internet (e.g.: Applets).

Java Environment

- Java Environment includes a large number of development tools and hundreds of classes and methods.
- The development tools are part of the system known as Java Development Kit (JDK) and the classes and methods are part of the Java Standard Library (JSL), also known as Application Programming Interface (API).
- JDK comes with a collection of tools that are used for developing and running java programs:
  - appleviewer (for viewing java applets)
  - javac (java compiler)
  - java (java interpreter)
  - javap (java disassembler)
  - javah (for C header files)
  - javadoc (for creating HTML documents)
  - jdb (Java debugger)

Java API

- It includes hundreds of classes and methods grouped into several functional packages.
- Most commonly used packages are:
  - Language Support Package: a collection of classes and methods required for implementing basic features of java.
  - Utilities Package: a collection of classes to provide utility functions such as date and time functions.
  - Input/output Package: a collection of classes required for input/output manipulation.
  - Networking Package: a collection of classes for communicating with other computers via Internet.
  - AWT Package (Abstract Window Tool kit package): contains classes that implements platform-independent graphical user interface.
  - Applet Package: includes set of classes that allows us to create java applets.

Programming Structure

- In Java programming language:
  - A program is made up of one or more *classes*
  - A class contains one or more *methods*
  - A method contains program *statements*
- In Java, first we need to import the required packages. By default, java.lang.* is imported. Java has several such packages in its library.
- A **package** is a kind of directory that contains a group of related classes and interfaces. A class or interface contains methods.
- Since Java is purely an Object Oriented Programming language, we cannot write a Java program without having at least one class or object. So, it is mandatory to write a class in Java program. We should use **class** keyword for this purpose and then write class name.

Contd…


```
//  comments about the class
public class Class-name
{
    //  comments about the method
    public static void main (String[] args)
    {

    }

}
```

class header

method header

method body

class body

Comments can be placed almost anywhere

First Java Program

```
// MyProgram.java
public class Sample
{
    public static void main( String args[])
    {
        System.out.println(" Hello World! ");
    }
}
```

Let us discuss the program line by line:

- Class Declaration: the first line class Sample declares a class, which is an object constructor. **Class** is keyword and declares a new class definition and **Sample** is a java identifier that specifies the name of the class to be defined.

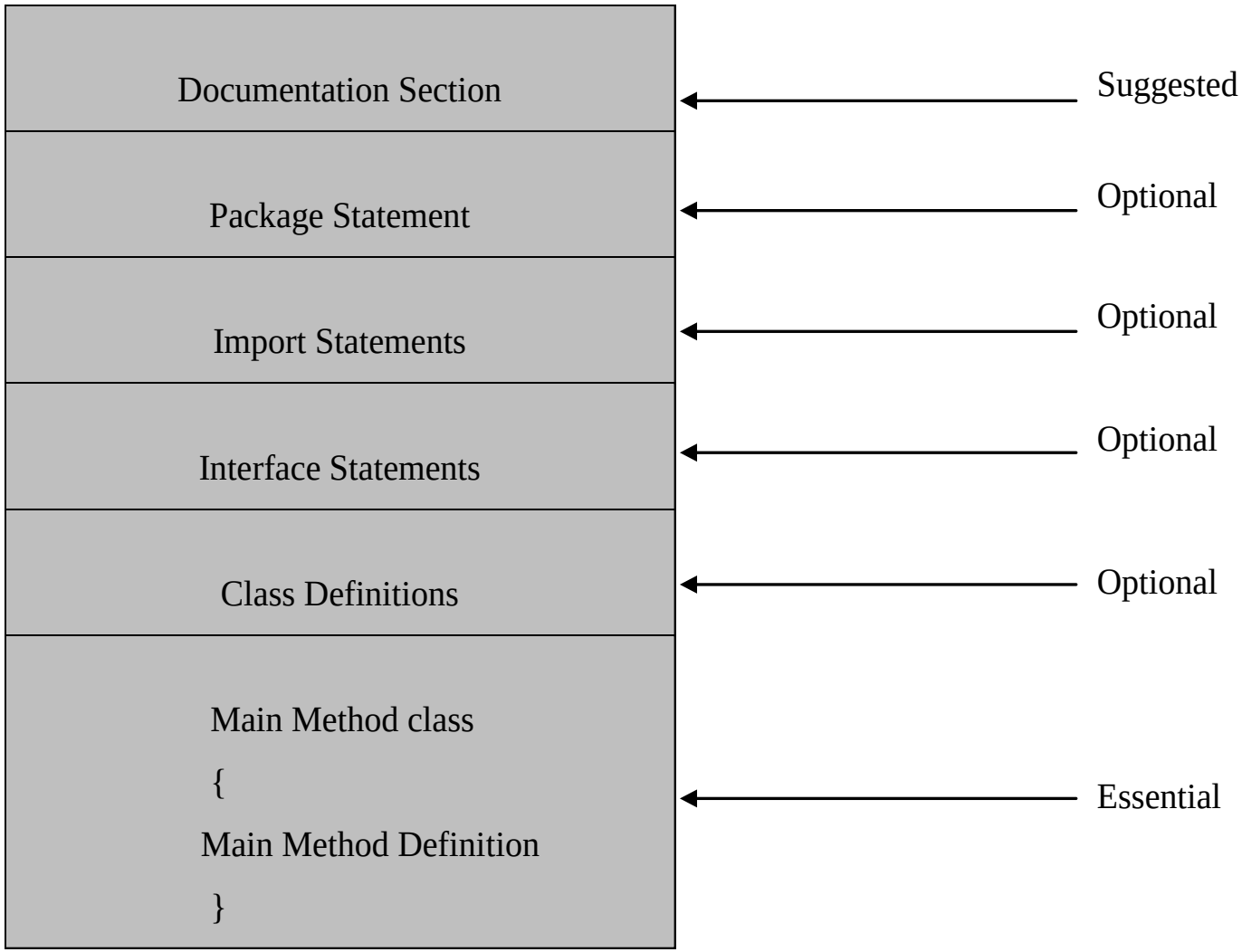
Contd…

- Braces : Every class definition in java begins with an opening brace "{" and ends with a closing brace "}".
- The main line: the third line *public static void main(String args[])* defines a method named as ***main,*** is the starting point for the interpreter to begin the execution of the program.
  - ***public****:* is an access specifier that declares the main method as "unprotected" and therefore making it accessible to all other classes.
  - ***static****:* declares that this method as one that belongs to the entire class and not a part of any objects of the class.
    - Main must always be declared as static since the interpreter uses this method before any objects are created.
  - ***void*** *:* states that the main method does not return any value (but prints some text to the screen.)

Contd…

- All parameters to a method are declared inside a pair of parenthesis. Here, **String args[ ]** declares a parameter named **args**, which contains array of objects of the class type **String**.
- The Output Line: The only executable statement in the program is **System.out.println("Hello World!");**
  - This is similar to cout<< constructor of C++.
  - The **println** method is a member of the ***out*** object, which is a static data member of **System** class.

Java Program Structure



Documentation Section

- Comprises a set of comment lines giving the name of the program, the author and other details, which the programmer would like to refer at a later stage.
- Comments are description about the aim and features of the program.
- Comments increase readability of a program.
- Java supports three types of comments:

1. Single line comment //
2. Multiple line comment /*……
   ……*/
3. Documentation comment /**....*/
   - This form of comment is used for generating documentation automatically.

Package Statement

- Is the first statement in Java file and is *optional*.
- It declares a package name and informs the compiler that the classes defined here belong to this package.

  Example: ***package student;***

Import statements

- Next to package statements (but before any class definitions) a number of import statements may exist. This is similar to #include statements in C or C++.
- Using import statements we can have access to classes that are part of other named packages.

  Example: ***import java.lang.Math;***

Interface Statements

- An interface is like a class but includes a group of method declarations.
- is also an optional section.
- is used only when we wish to implement the multiple inheritance features in the program

Class Definitions

- A Java program may contain multiple class definitions.
- Classes are the primary and essential elements of a Java program.
- These classes are used to map objects of real-world problems.
- The number of classes depends on the complexity of the problem.

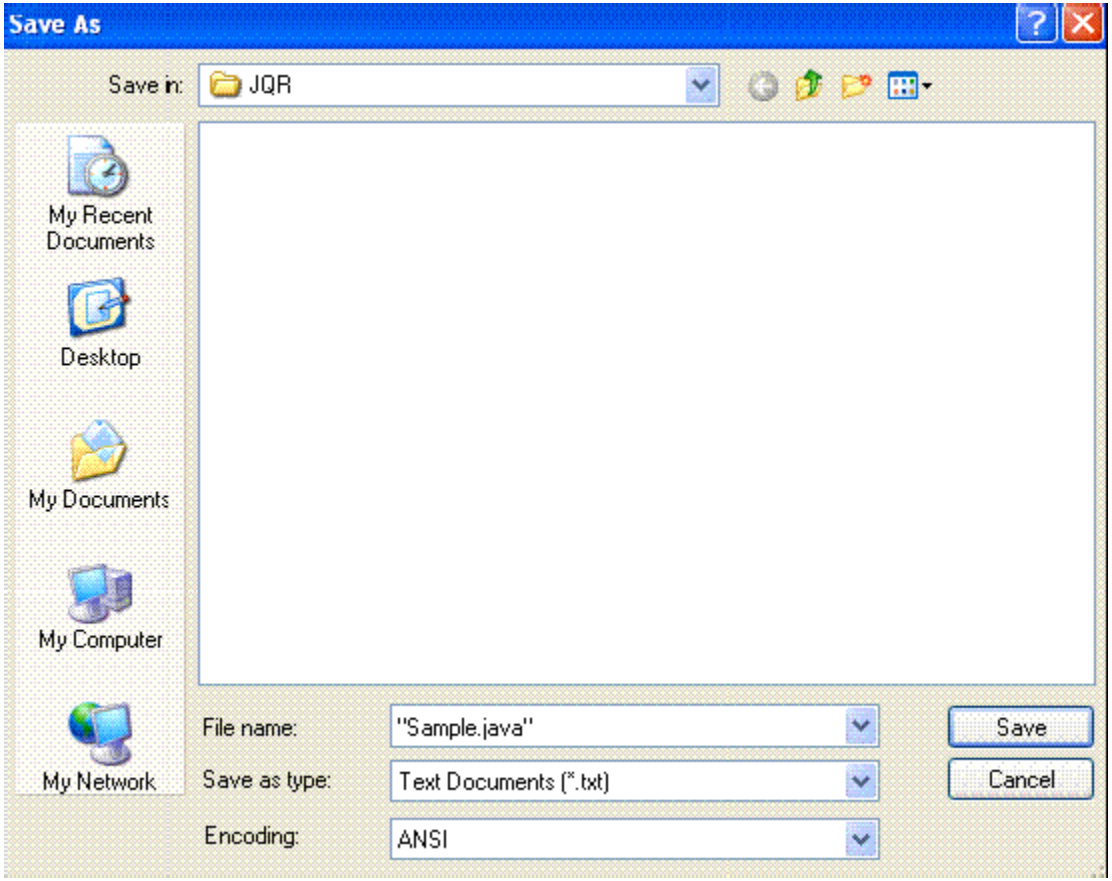
Main Method

- Since every Java stand-alone program requires a main method as its starting point, this class is the essential part of a Java program.
- A simple Java program may contain only this part.
- The main method creates objects of various classes and establishes communications between them.
- On reaching the end of main, the program terminates and control passes back to the operating system.

Creating a Source File

- Type the program in a text editor (i.e. Notepad, WordPad, Microsoft Word or Edit Plus).
  - We can launch the Notepad editor from the **Start menu by selecting Programs > Accessories > Notepad.** In a new document, type the above code (i.e. Sample Program).
- Save the program with filename same as Class_name (i.e. Sample.java) in which main method is written. To do this in Notepad, first choose the **File > Save** menu item. Then, in the **Save** dialog box:
- Using the **Save in** combo box, specify the folder (directory) where you'll save your file.
- In this example, the directory is JQR on the D drive.
- In the **File name** text field, type "Sample.java", including the quotation marks. Then the dialog box should look like this:

Contd…



Now click **Save**, and exit Notepad**.**

Executing the Source File

- To Compile the Sample.java program go to DOS prompt. We can do this from the **Start** menu by choosing **Run...** and then entering cmd. The window should look similar to the following figure.

- The prompt shows current directory. To compile Sample.java source file, change current directory to the directory where Sample.java file is located. For example, if source directory is JQR on the D drive, type the following commands at the prompt and press Enter:

- Now the prompt should change to D:\JQR>

Contd…

- At the prompt, type the following command and press Enter: **javac Sample.java**

- The compiler generates byte code and **Sample.class** will be created.
- To run the program, enter java followed by the class name created at the time of compilation at the command prompt in the

- The program interpreted and the output is displayed.

Java Tokens

- A class in java is defined by a set of declaration statements and methods containing executable statements.
- Most statements contain expressions, which describe the actions carried out on data.
- Smallest individual units in a program are known as ***tokens.***
- In simplest terms, a java program is a collection of ***tokens***, ***comments***, and ***white spaces.***
- Java has five types of tokens: *Reserved Keywords*, *Identifiers*, *Literals*, *Separators* and *Operators* .

1. Keywords

Java Programming

- Are essential part of a language definition and can not be used as names for variables, classes, methods and so on.
- Java language has reserved 60 words as keywords.

abstract	boolean	break	byte	byvalue*	case
cast*	catch	char	class	const*	continue
default	do	double	else	extends	false**
final	finally	float	for	future*	generic*
goto	if	implements	import	inner*	instanceof
int	interface	long	native	new	null**
operator*	outer*	package	private	protected	public
rest*	return	short	static	super	switch
synchronized	this	threadsafe	throw	throws	transient

2. Identifiers

- Are programmer-designed tokens.
- Are used for naming classes, methods, variables, objects, labels, packages and interfaces in a program.
- Java identifiers follow the following rules:
  - They can have alphabets, digits, and the underscore and dollar sign characters.
  - They must not begin with a digit
  - Uppercase and lowercase letters are distinct.
  - They can be of any length.

POP QUIZ

- Which of the following are valid Identifiers?

1) $amount
2) 6tally
3) my*Name
4) salary
5) score
6) first Name
7) total#
8) cast

3. Literals

- Literals in Java are a sequence of characters(digits, letters and other characters) that represent constant values to be stored in variables.
- Five major types of literals in Java:

I. Integer Literals: refers to a sequence of digits (*decimal integer, octal integer and hexadecimal integer*)

II. Floating-point Literals

III. Character Literals

IV. String Literals

V. Boolean Literals

4. Separators

- Are symbols used to indicate where groups of code are divided and arranged.
- They basically define the shape and functions of our code.
- Java separators include:

I. Parenthesis ( ) :- used to enclose parameters, to define precedence in expressions, surrounding cast types

II. Braces { } :- used to contain the values of automatically initialized arrays and to define a block of code for classes, methods and local scopes.

Contd…

III. Brackets [ ] :- are used to declare array types and for dereferencing array values.

IV. Semicolon ; :- used to separate statements.

V. Comma , :- used to separate consecutive identifiers in a variable declaration, also used to chain statements together inside a "*for*" statement.

VI. Period . :- Used to separate package names from sub-package names and classes; also used to separate a variable or method from a reference variable.

5. Operators

- Are symbols that take one or more arguments (operands) and operates on them to a produce a result.
- Are used to in programs to manipulate data and variables.
- They usually form a part of mathematical or logical expressions.
- Expressions can be combinations of variables, primitives and operators that result in a value.

Java Operators

- There are 8 different groups of operators in Java:
  - Arithmetic operators
  - Relational operators
  - Logical operators
  - Assignment operator
  - Increment/Decrement operators
  - Conditional operators
  - Bitwise operators
  - Special operators

A. Arithmetic Operators

- Java has five basic arithmetic operators

Operator	Meaning
+	Addition or unary plus
–	Subtraction or unary minus
*	Multiplication
/	Division
%	Modulo division

- They all work the same way as they do in other languages.
- We cannot use these operators on boolean type .
- Unlike C and C++, modulus operator can be applied to the floating point data.
- Order of operations (or precedence) when evaluating an expression is the same as you learned in school (BODMAS).

B. Relational Operators

- Relational operators compare two values
- Produces a boolean value (**true** or **false**) depending on the relationship.
- Java supports six relational operators:

Operator	Meaning
<	**Is less than**
<=	**Is less than or equal to**
>	**Is greater than**
>=	**Is greater than or equal to**
==	**Is equal to**
!=	**Is not equal to**

- Relational expressions are used in decision statements such as, *if and while* to decide the course of action of a running program.

Examples of Relational Operations

```
int x = 3;
int y = 5;
boolean result;
```

1) `result = (x > y);`

now `result` is assigned the value false because 3 is not greater than 5

2) `result = (15 == x*y);`

now `result` is assigned the value true because the product of 3 and 5 equals 15

3) `result = (x != x*y);`

now `result` is assigned the value true because the product of `x` and `y` (15) is not equal to `x` (3)

C. Logical Operators

Symbol	Name
&&	Logical AND
||	Logical OR
!	Logical NOT

- Logical operators can be referred to as `boolean` operators, because they are only used to combine expressions that have a value of `true` or `false`.
- The logical operators `&&` and `||` are used when we want to form compound conditions by combining two or more relations.
- An expression which combines two or more relational expressions is termed as a `logical expression` or a `compound relational expression`.

Examples of Logical Operators


```
boolean x = true;
boolean y = false;
boolean result;
```

1. Let `result = (x && y);`

   now `result` is assigned the value `false`
   (see truth table!)

2. Let `result = ((x || y) && x);`

   `(x || y)` evaluates to **`true`**
   `(true && x)` evaluates to **`true`**

   now `result` is assigned the value `true`

4. Assignment Operator

- Are used to assign the value of an expression to a variable.
- In addition to the usual assignment operator(=), java has a set of 'shorthand' assignment operators which are used in the form:

```
var op = expr ;
```

Where `var` is a variable, `op` is a binary operator and `expr` is an expression. The operator `op =` is known as shorthand assignment operator.

- The assignment statement: `var op= expr;` is equivalent to `var=var op(expr);`
- Examples:

`x += y + 5;` is equivalent to `x = x+(y+5);`

`y *= 7;` is equivalent to `y = y * 7;`

5. Increment/Decrement Operators

```
count = count + 1;
```

can be written as:

```
++count; or count++;
```

++ is called the increment operator.

```
count = count - 1;
```

can be written as:

```
--count; or count--;
```

-- is called the decrement operator.

- Both **++** and **--** are unary operators.

Increment/Decrement operator has two forms:

- The prefix form ++count, --count
  first adds/subtracts1 to/from the variable and then continues to any other operator in the expression

```
int numOranges = 5;
int numApples = 10;
int numFruit;
numFruit = ++numOranges + numApples;
numFruit has value 16
numOranges has value 6
```

- The postfix form count++, count--
  first evaluates the expression and then adds 1 to the variable

```
int numOranges = 5;
int numApples = 10;
int numFruit;
numFruit = numOranges++ + numApples;
numFruit has value 15
numOranges has value 6
```

6. Conditional Operators

- The character pair `?:` is a ternary operator available in java.
- It is used to construct conditional expression of the form:

```
exp1 ? exp2: exp3;
```

where `exp1`, `exp2` and `exp3` are expressions.

- The operator `?:` works as follows:
  - `exp1` is evaluated first. If it is nonzero (true), then the expression `exp2` is evaluated and becomes the value of the conditional expression. `exp3` is evaluated and becomes the value of the conditional expression if `exp1` is false.
- Example:

  Given `a=10`, `b=15` the expression

```
x=(a>b)? a:b; will assign the value of b to
x. i.e. x=b=15;
```

6. Bitwise Operators

- One of the unique features of java compared to other high-level languages is that it allows direct manipulation of individual bits within a word.
- Bitwise operators are used to manipulate data at values of bit level.
- They are used for testing the bits, or shifting them to the right or left.
- They may not be applied to float or double data types.

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
~	One's complement
<<	Shift left
>>	Shift right
>>>	Shift right with zero fill

7. Special Operators

- Java supports some special operators of interest such as `instanceof` operator and `member selection` operator(`.`).

`7.1 instanceof Operator:` is an object reference operator and returns true if the object on the left-hand side is an instance of the class given on the right-hand side.

Example : `person instanceof student;`

is true if the object person belongs to the class student; otherwise it is false

`7.2 Dot Operator:` is used to access the instance variables and methods of class objects.

Example:

```
person.age;          // Reference to the variable age.
person1.salary(); // Reference to the method salary.
```

Type Conversion in Expressions

A. Automatic Type Conversion: If the operands of an expression are of different types, the 'lower' type is automatically converted to the 'higher' type before the operation proceeds. That is, the result is of the 'higher' type.

Example :

```
int a=110;

float b=23.5;
```

the expression `a+b` yields a floating point number 133.5 since the 'higher' type here is float.

Contd…

B.Casting a value: is used to force type conversion.

- The general form of a cast is:

```
(type-name)expression;
```

where `type-name` is one of the standard data types.

Example :

`x=(int)7.5;` will assign 7 to x

`a=(int)21.3/(int)3.5;` a will be 7

Operator Precedence and Associativity

Operator	Description	Associativity	Rank
.	Member Selection	Left to right	1
( )	Function call		
[ ]	Array elements reference		
-	Unary Minus	Right to left	2
++	Increment		
--	Decrement		
!	Logical negation		
~	One's complement		
(type)	Casting		
*	Multiplication	Left to right	3
/	Division		
%	Modulus		

Contd…

Operator	Description	Associativity	Rank
+	Addition	Left to right	4
-	Subtraction		
<<	Left Shift	Left to right	5
>>	Right Shift		
>>>	Right shift with zero fill		
<	Less than	Left to right	6
<=	Less than or equal to		
>	Greater than		
>=	Greater than or equal to		
instanceof	Type comparison		
==	Equality	Left to right	7
!=	Inequality		

Contd…

Operator	Description	Associativity	Rank
&	Bitwise AND	Left to right	8
^	Bitwise XOR	Left to right	9
|	Bitwise OR	Left to right	10
&&	Logical AND	Left to right	11
||	Logical OR	Left to right	12
?:	Conditional Operator	Left to right	13
=	Assignment operator	Left to right	14
Op=	Shorthand assignment	Left to right	15

POP QUIZ

Java Programming

1) What is the value of number? -12

```
int number = 5 * 3 - 3 / 6 - 9 * 3;
```

2) What is the value of result? false

```
int x = 8;
int y = 2;
boolean result = (15 == x * y);
```

3) What is the value of result? true

```
boolean x = 7;
boolean result = (x < 8) && (x > 4);
```

4) What is the value of numCars? 27

```
int numBlueCars = 5;
int numGreenCars = 10;
int numCars = numGreenCars++ + numBlueCars +
++numGreeenCars;
```

Variables

- A variable is an identifier that denotes a storage location used to store a data value.
- Unlike constants, that remain unchanged during the execution of a program, a variable may take different values at different times during the execution of the program.
- It is good practice to select variable names that give a good indication of the sort of data they hold:
  - For example, if you want to record the size of a hat, `hatSize` is a good choice for a name whereas `qqq` would be a bad choice.

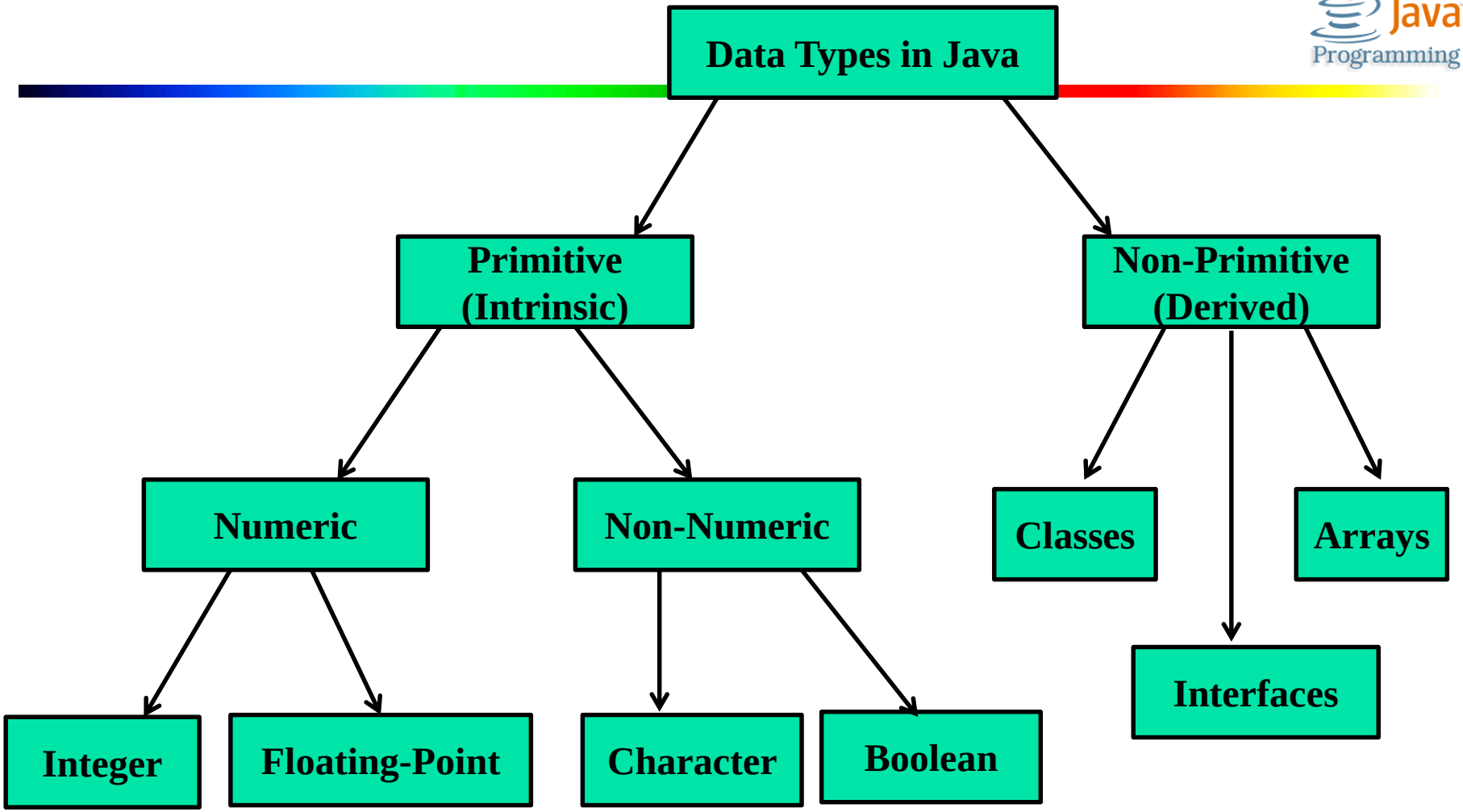
Contd…

- Variable names may consist of alphabets, digits, the underscore (_) and dollar ($) characters, subject to the following conditions:

  1. They should not begin with a digit.
  2. Keywords should not be used as a variable name.
  3. White spaces are not allowed.
  4. Uppercase and lowercase are distinct. i.e. A rose is not a Rose is not a ROSE.
  5. Variable names can be of any length.

Data Types

- Every variable in Java has a data type.
- Data types specify the size and type of values that can be stored.
- Java is language is rich in the data types.
- Java data types are of two type:
  - Primitive Data Types (also called intrinsic or built-in data types)
  - Non-Primitive data Types (also known as Derived or reference types)



Data Types in Java

Primitive Data Types

- There are eight built-in data types in Java:
  - 4 integer types (`byte`, `short`, `int`, `long`)
  - 2 floating point types (`float`, `double`)
  - Boolean (`boolean`)
  - Character (`char`)
- All variables must be declared with a data type before they are used.
- Each variable's declared type does not change over the course of the program.

A. Integer Data types

- There are four data types that can be used to store integers.
- The one you choose to use depends on the size of the number that we want to store.

Data Type	Description
`byte`	Variables of this kind can have a value from: **-128 to +127** and occupy 8 bits in memory
`short`	Variables of this kind can have a value from: **-32768 to +32767** and occupy 16 bits in memory
`int`	Variables of this kind can have a value from: **-2147483648 to +2147483647** and occupy 32 bits in memory
`long`	Variables of this kind can have a value from: **-9223372036854775808 to +9223372036854775807** and occupy 64 bits in memory

B. Floating-Point Types

- Integer types can hold only whole numbers and therefore we need another type known as floating point type to hold numbers containing fractional parts.
- There are two data types that can be used to store decimal values (real numbers).

Data Type	Description
`float`	Variables of this kind can have a value from: **1.4e(-45) to 3.4e(+38)**
`double`	Variables of this kind can have a value from: **4.9e(-324) to 1.7e(+308)**

C. Character Type

- Is used to store character constants in memory.
- Java provides a character data type called ***char***
- The char data type assumes a size of 2 bytes but, basically, it can hold only a single character.
- Note that you need to use singular quotation marks while initializing a variable whose data type is `char`.

Example:

```
char firstLetterOfName = 'e' ;
char myQuestion = '?' ;
```

D. Boolean Type

- Boolean is a data type used when we want to test a particular condition during the execution of the program.
- There are only two values that a Boolean can take: **`true`** or **`false`**.
- Boolean type is denoted by the keyword **`boolean`** and uses only one bit of storage.
- All comparison operators return boolean type values.
- Boolean values are often used in selection and iteration statements.

Declaration of Variables

- After designing suitable variable names, we must declare them to the compiler. Declaration does three things:
  1. It tells the compiler what the variable name is
  2. It specifies what type of data the variable will hold
  3. The place of declaration (in the program) declares the scope of the variable.
- A variable must be declared before it is used in the program.
- The general form of declaration of Variables is:

```
type variable1, variable2,...., variableN;
```

Example:

```
int count, x,y;                          //Declaration
char firstLetterOfName = 'e' ;  // Declaration & initialization
```

Assigning Values to Variables

- A variable must be given a value after it has been declared but before it is used in an expression in two ways:
  - By using an assignment statement
  - By using a read statement

Assignment Statement

- A simple method of giving value to a variable is through the assignment statement as follows:

```
variableName = value;
```

Example:

```
x = 123, y = -34;
```

- It is possible to assign a value to a variable at the time of declaration as:

```
type variableName = value;
```

Read Statement

- It is also to assign value for variables interactively through the keyboard using the *readLine()* method which belongs to the *DataInputStream* class.
- The *readLine()* method reads the input from the keyboard as a string which is then converted into the corresponding data type using the data type wrapper classes.
- The wrapper classes are contained in the *java.lang* package.
- Wrapper classes wrap a value of the primitive types into an object.
- The keywords *try* and *catch* are used to handle any errors that might occur during the reading process.

```
// A program to read data from the Keyboard
import java.io.DataInputStream;
public class Reading{
 public static void main(String[] args)
{
      DataInputStream in = new DataInputStream(System.in);
      int intNumber=0;
      float floatNumber=0.0f;
      double doubleNumber=0.0;
      String Name=null;
     try{
         System.out.println("Enter your Name: ");
         Name=in.readLine();
         System.out.println("Enter an Integer Number: ");
         intNumber=Integer.parseInt(in.readLine());
         System.out.println("Enter a float Number: ");
         floatNumber=Float.parseFloat(in.readLine());
         System.out.println("Enter a Double Number Number: ");
         doubleNumber=Double.parseDouble(in.readLine());
      }
      catch(Exception e){}
     System.out.println("Hello : "+Name);
     System.out.println("The Integer Number is : "+intNumber);
     System.out.println("The Float Number is : "+floatNumber);
     System.out.println("The Double Number is : "+doubleNumber);
   }
}
```

Scope of Variables

1. **Instance Variables:** are declared in a class, but outside a method, constructor or any block.
   - are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.
   - They take different values for each object

2. **Class Variables:** are also known as static variables, are declared with the *static* keyword in a class, but outside a method, constructor or a block.
   - Are global to a class and belong to the entire set of objects that class creates.
   - Only one memory location is created for each class variable.

3. **Local Variables:** are variables declared and used inside methods.
   - Can also be declared inside program blocks that are define between **{** and **}**.