Chapter 3

Values and Data Types

Declaration

- These slides are made for UIT, BU students only. I am not holding any copy write of it as I had collected these study materials from different books and websites etc. I have not mentioned those to avoid complexity.

Topics

- The Character Set
- Character encoding
  - ASCII
  - Unicode
- Escape sequence
- Tokens
- Reserved Keywords
- Identifiers
- Literals or Constants
- Operators
- Separators
- Variables

Topics

- Data types
- Primitive Data types
  - Integers
  - Floating points
  - Characters
  - Boolean
- Non-Primitive Data types
- Character Escape Sequences
- Scope and lifetime of the variable
- Java is strongly typed language
- Type Conversion and casting
- Java's Automatic Conversions
- Casting incompatible types
- Arrays

The Character Set

- Like any other human readable language, programming languages also use Characters, Words (tokens) and Sentences (statements).
- The character set is a set of alphabets, digits and some special characters that are valid in Java language.
- The smallest unit of Java language is the set of characters needed to write Java tokens. These character set are defined by Unicode character set.
- In Java Programming, each and every character is considered as a single lexeme. i.e., Basic Lexical Element. Each Java Program is set of statements and each statement is set of different Java programming lexemes.

The Character Set

- A character set consists of Alphabets, Digits and Special Characters
- Alphabets
  - Uppercase: ABC ....XYZ
  - Lowercase: abc……xyz
- Digits
  - 0 1 2 3 4 5 6 7 8 9
- Special Characters
  - + - () {} [ ] \ | / > < etc.
- However, we all know that a computer can only understand binary language that is just a sequence of 0s and 1s. So how is it going to understand these alphabets, digits and symbols? Here comes the concept of encoding.

Character encoding

- A character encoding tells the computer how to interpret raw zeroes and ones into real characters by pairing numbers with characters. It’s just like saying let’s give a number to capital ‘A’ that is 65 and ‘B’ 66 and so on up to Z. There can be similar corresponding numbers to different letters, digits and symbols. Now these numbers can easily be converted into binary number.
- Words and sentences in text are created from characters.
- There are many different types of character encodings, such as ASCII, 8-bit encodings, and Unicode-based encodings.
- The Unicode standard defines Unicode Transformation Formats (UTF) UTF-8, UTF-16, and UTF-32, and several other encodings.

ASCII

- ASCII (American Standard Code for Information Interchange) Is a Character-encoding scheme for representing English characters as numbers, with each letter is assigned a number from 0 to 127. It was the first character encoding standard.
- In an ASCII file, each alphabetic, numeric, or special character is represented with a 7-bit binary number (a string of seven 0s or 1s). 128 possible characters are defined.
- UNIX and DOS-based operating systems use ASCII for text files.
- Based on the ASCII table, when we store 'TAB' and decimal 9 to the memory, they are both stored as "1001". How does the computer know it's a 'TAB' or a decimal 9?
- The computer doesn't "know" what type a specific address in memory is, that knowledge is baked into the instructions of your program.

Unicode

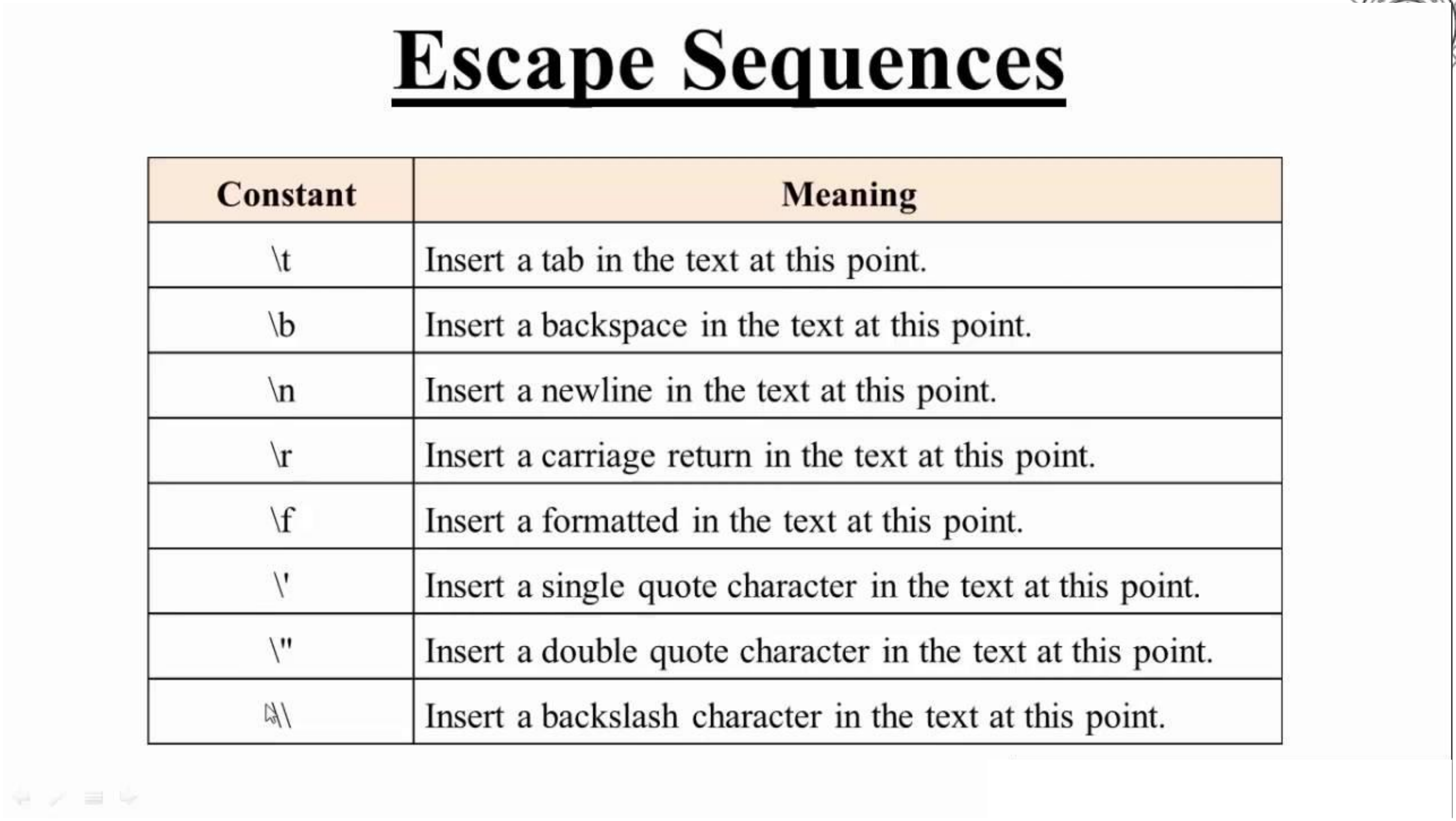
- The Unicode Standard provides a unique number for every character, irrespective of platform, device, application or language. It has been adopted by all modern software providers allowing to be transported through many different platforms, devices and applications without corruption.
- Unicode forms the foundation for the representation of languages and symbols in all major operating systems, search engines, browsers, laptops, and smartphones plus the Internet and World Wide Web (URLs, HTML, XML, CSS, JSON, etc.).
- Currently, the Unicode standard contains 34168 distinct coded characters derived from 24 supported language scripts. These characters cover the principal written languages in the world.

Unicode

- The Unicode is a 16-bit (2 byte) character set which can represent almost all human alphabets and writing systems around the world. This character set is used is Java programming language.

Escape sequence

- Escape characters are used to signal an alternative interpretation of a series of characters. They are also called escape sequences or escape codes.
- In Java, a character preceded by a backslash (\) is an escape sequence and has special meaning to the Java compiler.



Escape Sequences

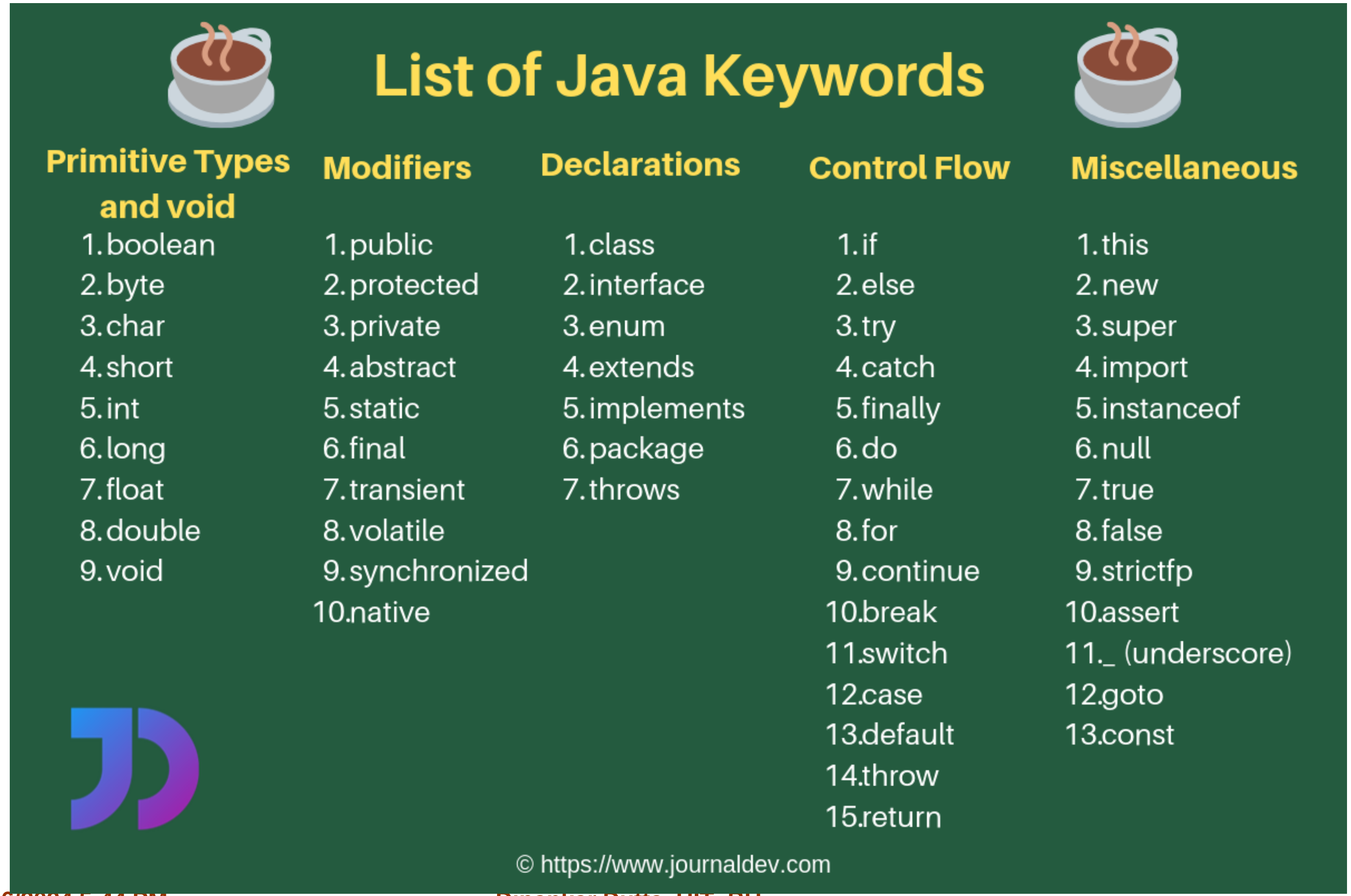
Constant	Meaning
\t	Insert a tab in the text at this point.
\b	Insert a backspace in the text at this point.
\n	Insert a newline in the text at this point.
\r	Insert a carriage return in the text at this point.
\f	Insert a formatted in the text at this point.
\'	Insert a single quote character in the text at this point.
\"	Insert a double quote character in the text at this point.
\\	Insert a backslash character in the text at this point.

Tokens

- In every human language (like English), there are words and symbols which have a predefined meaning. Similarly, all programming languages have a set of predefined words and symbols. In Java programming language we call them tokens.
- There are five types of tokens in Java
  - Reserved Keywords
  - Identifiers
  - Literals or constants
  - Operators
  - Separators

Reserved Keywords

- Keywords are words that have already been defined for Java compiler. They have special meaning for the compiler.



List of Java Keywords

Primitive Types and void	Modifiers	Declarations	Control Flow	Miscellaneous
1. boolean	1. public	1. class	1. if	1. this
2. byte	2. protected	2. interface	2. else	2. new
3. char	3. private	3. enum	3. try	3. super
4. short	4. abstract	4. extends	4. catch	4. import
5. int	5. static	5. implements	5. finally	5. instanceof
6. long	6. final	6. package	6. do	6. null
7. float	7. transient	7. throws	7. while	7. true
8. double	8. volatile		8. for	8. false
9. void	9. synchronized		9. continue	9. strictfp
	10. native		10. break	10. assert
			11. switch	11. _ (underscore)
			12. case	12. goto
			13. default	13. const
			14. throw	
			15. return	

© https://www.journaldev.com

Identifiers

- Identifiers represent names which can be assigned to variables, methods and classes to uniquely identify them.
- In Java the user (programmer) defined names are called identifiers.
- An Example:

```
public class Test
{
        public static void main (String[] args)
        {
                int a=20;
        }
}
```

Identifiers

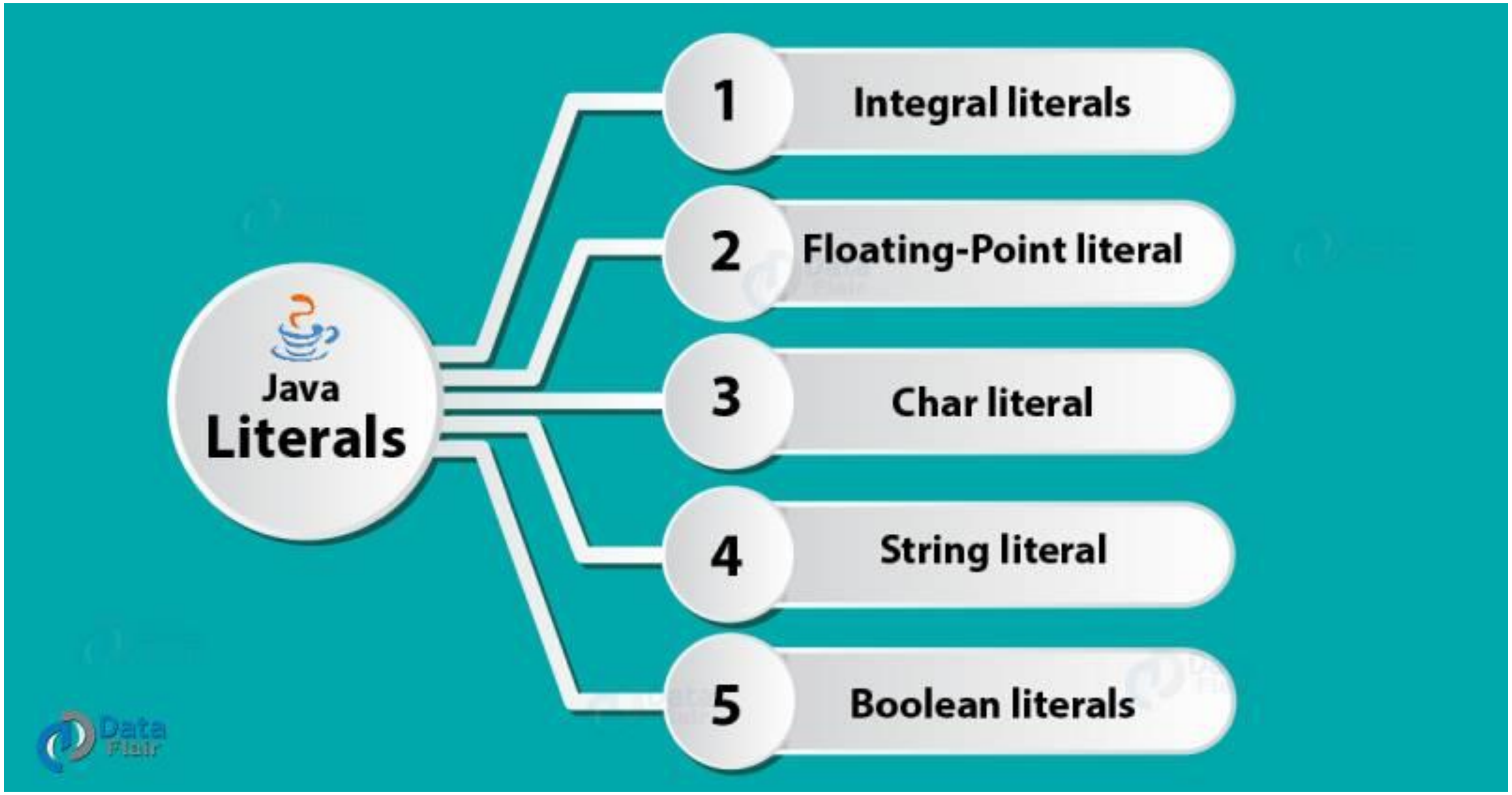
- In the above java code, we have 4 identifiers namely:
  - Test: class name.
  - main: method name
  - args: variable name
  - a: variable name

Identifiers

- An identifier is a sequence of characters that consist of letters, digits, underscores (_), and dollar signs ($).
- An identifier must start with a letter, an underscore (_), or a dollar sign ($). It cannot start with a digit.
- An identifier cannot be a reserved word.
- An identifier cannot be true, false, or null.
- An identifier can be of any length.

Literals or Constants

- A literal or constant, in Java, refers to a fixed value that does not change during execution of the program. Java supports several types of constants.



Literals or Constants

1. Integer constant: Integer constant refers to sequence of numbers without decimal point. There are three types of integer constants

A. Decimal integers (base 10): They consist of combination of digits 0 through 9 with or without negative number This is the number system you use every day.
   - Example: 105   -245   0

B. Octal integer (base 8) : They consist of combination of digits 0 through 7 with leading Zero.
   - Example: 074   0   034

C. Hexadecimal integer (base 16) : They consist of combination of digits 0 through 9 and letter a through A to F with leading (zero x) 0x or (zero X) 0X.
   - Example: 0X7A4   0XA   0xA4

Literals or Constants

2. Real constants: It refers to sequence of numbers with a decimal point. They can be represented in two forms: decimal notation and exponential notation.

A. Decimal notation: The decimal notation has three parts the integer part, the decimal point and the fractional part and can be used to represent numbers.

- Example:

Valid Real Constants 2.15 -7.8 0.78 -111.55555

Invalid Real Constants

- 9. (No numeral after decimal point)
- 99.320.34 (Two decimal points)
- 15,55.890 (Comma not allowed)

Literals or Constants

B. Exponential notation: The exponential notation is used to represent numbers and has the following syntax

mantissa e/E exponent

The mantissa is either a real number expressed in decimal notation or an integer.

The exponent is an integer with an optional + or - sign. Thus e2 means 10^2.

Example:

- Valid Real Constants in exponent form

0:97e4 12e-2 -1.2E-4 10.10E+5 10.E+5f .25e7

When representing a **float** data type in **Java**, we should append the letter **f** to the end of the data type; otherwise it will save as double.

- Invalid Real Constants in exponent form
  - 2.5E (No numerals specified after exponent)
  - 10.5E4.8 (Exponent can not have decimal part}
  - 20,555E4 (Comma not allowed)

Literals or Constants

3. Character constant: A single character enclosed within single quotes are called character constant.
   - Example: ‘5’ ‘E’ ‘$’ ‘+ ‘
4. String Constants: A sequence of characters enclosed with double quotes are known as String Constants.
   - Example: “Hai” “Java programming” “1998”
5. Boolean constants : The Boolean constants can have only two types of value i.e., true or false. True and false are the reserved words of Java language.

Operators

- Java operators are the mathematical symbols used to perform mathematical and logical operations among two or more variables or literals called operands. There are many type of operators available in java such as Arithmetic Operators, Relational Operators, Logical Operators, bitwise Operators, Assignment Operators etc.

Separators

- Separators help define the structure of a program. The separators used in Java are as follows.

Separator	Name	Use
.	Period	It is used to separate the package name from sub-package name & class name. It is also used to separate variable or method from its object or instance.
,	Comma	It is used to separate the consecutive parameters in the method definition. It is also used to separate the consecutive variables of same type while declaration.
;	Semicolon	It is used to terminate the statement in Java.
()	Parenthesis	This holds the list of parameters in method definition. Also used in control statements & type casting.
{}	Braces	This is used to define the block/scope of code, class, methods.
[]	Brackets	It is used in array declaration.

Variables

- Variable is the name of reserved area allocated in memory.
- The amount of memory allocated to a variable is defined by the type of data it holds.
- We explicitly define the data type by using some keywords of data types in Java.
- The value of the variable can change.
- The variable name is defined by the naming rules of identifiers, as we have already studied that a variable is one of the identifiers.

Variables

- The general form of variable declaration is given below:

  type identifier [= value];

- Where, the type is one of Java's primitive data types, or the name of a class or interface.
- The identifier is the name of the variable.
- The variable can be initialized by assigning a value of compatible type using assignment operator.
- More than one variable of a type can be declared using comma operator.

Variables

int a, b, c; // Declares three ints, a, b, and c.

int a = 10, b = 10; // Example of initialization

byte B = 22; // initializes a byte type variable B.

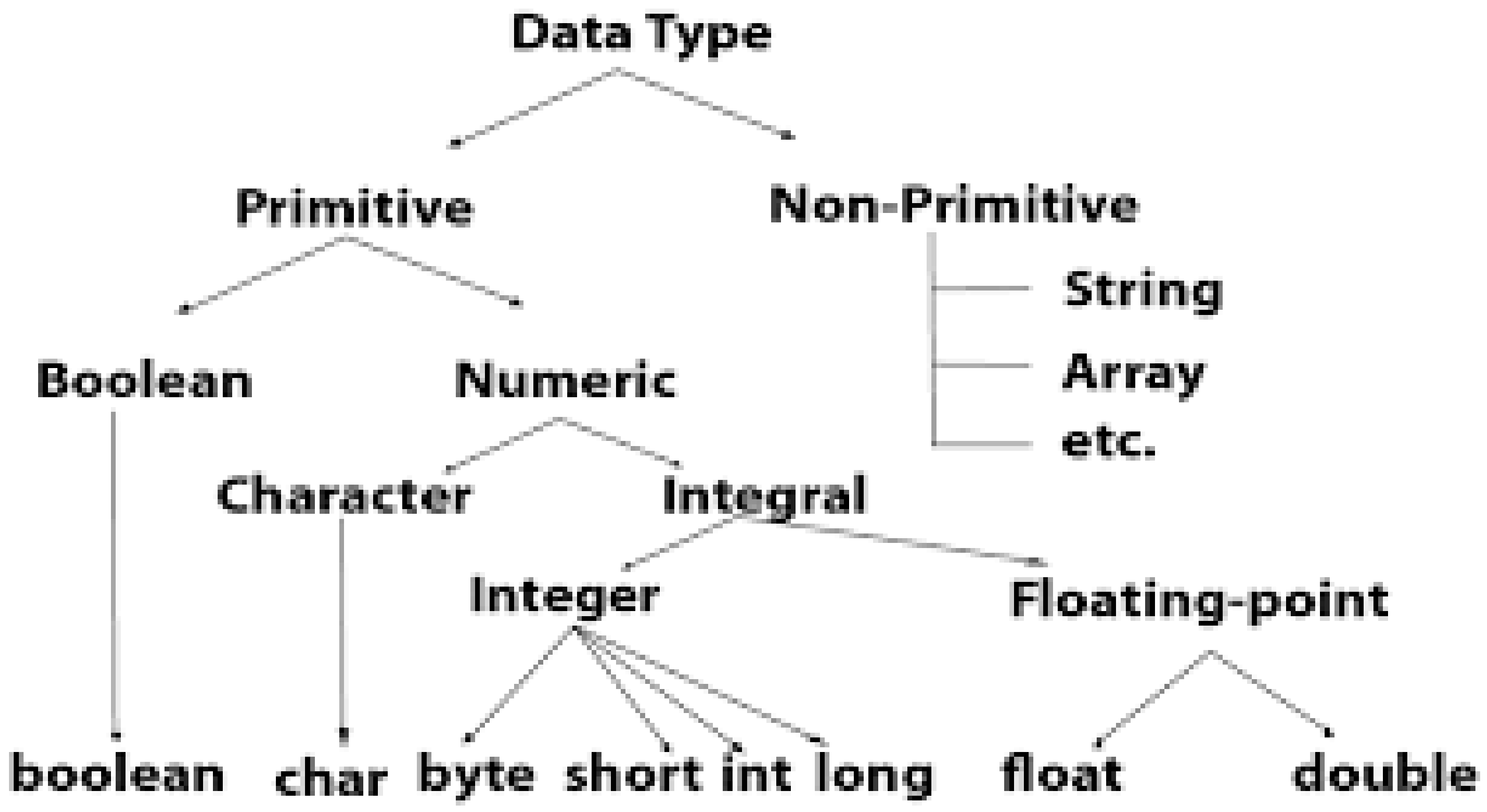
double pi = 3.14159; // declares and assigns a value of PI.

char a = 'a'; // the char variable a is initialized with value 'a'

Data types

- While declaring a variable we have to tell the Java compiler how much space this variable will occupy. For this purpose data types are used as different data types need different amount of memory space.
- A data type is defined as the set of possible values a variable can hold.
- In Java there are basically two types of data types.
  - Primitive data type
  - Non-primitive data type

Data types


Data Type
Primitive
Non-Primitive
String
Array
etc.
Boolean
Numeric
Character
Integral
Integer
Floating-point
boolean
char
byte
short
int
long
float
double

Primitive Data types

- Primitive Data types or Fundamental Data types are predefined data type provided by any programming language to store data in the variable.
- For Java, they can be classified into four groups.
  1. Integers
  2. Floating-Point numbers
  3. Characters
  4. Boolean

Primitive Data types: Integer

- Integers are whole numbers with no fractional part, such 45 and 0.
- Java does not support unsigned, positive only integers.
- This group includes byte, short, int, and long data types.

Name	Width	Range
long	64	-2^{63} to $2^{63}-1$
int	32	-2^{31} to $2^{31}-1$
short	16	-2^{15} to $2^{15}-1$
byte	8	-2^{7} to $2^{7}-1$

Primitive Data types: Floating points

- A number having fractional part is a floating-point number. For example 3.14, 2.0, -10.5 are floating-point numbers.
- The floating point numbers can be written in decimal notation or in exponential notation.
- For example, 179.5102 can be written as 1.795101E02.

Name	Width	Range
double	64	-1.7e – 308 to 1.7e + 308
float	32	-3.4e – 038 to 3.4e + 038

- float a = 5.555f; if it is written as float a = 5.555 will give rise to compile time error as any real value like 5.555 is a double data type to make it float you have to make use of suffix f or F.

Primitive Data types: Characters

- In Java, the datatype used to store characters is char.
- Java uses Unicode to represent characters.
- Example: char c1=‘Z’

Primitive Data types: Boolean

- Java has a simple type, called boolean, for logical values.
- It can have only two possible values, true or false.
- The boolean type is declared using the boolean keyword.
- The program code given below demonstrates the boolean type.
- For example: boolean b1= true;

Primitive Data types: Boolean

- It can have to possible values: true and false.

```
// Demonstrate boolean values.
class BoolTest {
  public static void main(String args[]) {
    boolean b;
    b = false;
    System.out.println("b is " + b);
    b = true;
    System.out.println("b is " + b);
    // a boolean value can control the if statement
    if(b) System.out.println("This is executed.");
    b = false;
    if(b) System.out.println("This is not executed.");
    // outcome of a relational operator is a boolean value
    System.out.println("10 > 9 is " + (10 > 9));
  }
}
```

Standard default values

Type	Default value
byte	zero:(byte)0
short	zero:(short)0
int	zero:0
long	zero:0L
float	0.0f
double	0.0d
char	null character
boolean	false

Non-Primitive Data types

- Non-primitive data type are the data type made up with the combinations of multiple variables of primitive data types. Non-primitive data types are Arrays, Strings and other complex data structure.

Non-Primitive Data types: String

- We have learnt how to create character variables. But we may need to store a word in a variable.
- The String type is used to declare a string variable.
- String defines an object of class String.
- A quoted string constant can be assigned to String variable.
- Java supports String to allow it to act similar to a primitive data type.
- For Example: String str= “ABC”;
- In C/C++ strings are implemented as array of characters.
- In Java string is a object types.

Character Escape Sequences

Escape Sequence	Description
\ddd	Octal character (ddd)
\uxxxx	Hexadecimal Unicode Character(xxxx)
\’	Single quote
\”	Double quote
\\	Backslash
\r	Carriage return
\n	New line
\f	Form feed
\t	Tab
\b	Backspace

Scope and lifetime of the variable

- The object declared in the outer scope will be visible to code within the inner scope.
- Reverse is not true.
- Lifetime of a variable is confined to its scope.

Scope and lifetime of the variable

```
// Demonstrate block scope.
class Scope {
  public static void main(String args[]) {
    int x; // known to all code within main
    x = 10;
    if(x == 10) { // start new scope
      int y = 20; // known only to this block
      // x and y both known here.
      System.out.println("x and y: " + x + " " + y);
      x = y * 2;
    }
    // y = 100; // Error! y not known here
    // x is still known here.
    System.out.println("x is " + x);
  }
}
```

Scope and lifetime of the variable

- One cannot declare a variable to have the same name as one in an outer scope.
- Here Java differs from C/C++.

```
// This program will not compile
class ScopeErr {
   public static void main(String args[]) {
     int bar = 1;
     {              // creates a new scope
       int bar = 2; // Compile time error -- bar already defined!
     }
   }
}
```

Java is strongly typed language

1. Every variable has a type, every expression has a type, and every type is strictly defined.
2. All assignments, whether explicit or via parameter passing in method calls, are checked for type compatibility. There are no automatic conversions of the conflicting types as in some languages.

- Part of java's safety and robustness comes from this fact.
- In C/C++ floating point value can be assign to an integer. In Java, it is not possible.

Type Conversion and casting

- If two variable types are compatible, then Java will perform the conversion automatically.
- For example, it is always possible to assign an int value to a long variable.
- Not all type conversion are implicitly allowed.
- To obtain conversion between incompatible types, one have to do casting.

Java's Automatic Conversions

- Two conditions
  - The two types are compatible
  - The destination types is longer than the source type.
- This is widening conversion.
- For example the int type is always large enough to hold all valid byte values. No explicit cast statement is required.
- Numeric types (integer and floating-point types) are compatible with each other.
- Numeric types are not compatible with char or boolean.
- char and boolean are not compatible with each other.

Casting incompatible types

- If someone want to convert int value to a byte variable.
- This conversion will not be performed automatically.
- Because byte is smaller than int.
- This type of conversion is called narrowing conversion.
- General form: (target-type)value

```
int a;
byte b;
// ...
b = (byte) a;
```

- It will reduced modulo (the remainder of an integer division by).

Casting incompatible types

- A floating point value is assigned to an integer type: truncation.

```
// Demonstrate casts.
class Conversion {
  public static void main(String args[]) {
    byte b;
    int i = 257;
    double d = 323.142;

    System.out.println("\nConversion of int to byte.");
    b = (byte) i;
    System.out.println("i and b " + i + " " + b);
```

Casting incompatible types

```
System.out.println("\nConversion of double to int.");
    i = (int) d;
    System.out.println("d and i " + d + " " + i);

    System.out.println("\nConversion of double to byte.");
    b = (byte) d;
    System.out.println("d and b " + d + " " + b);
  }
}
```

Arrays

- Arrays in Java work differently then they do in other languages.
- General form of declaration
- Type var_name[];
- int month_days[];
- Value of month_days is set to null.
- Now var_name=new type[size];
- new is a special operator that allocates memory.
- Type specified the type of data being allocated.
- Size specifies number of elements in the array.
- The elements will automatically initialized to zero.

Arrays

```
// Demonstrate a one-dimensional array.
class Array {
 public static void main(String args[]) {
  int month_days[];
  month_days = new int[12];
  month_days[0] = 31;
  month_days[1] = 28;
  month_days[2] = 31;
  …
  month_days[9] = 31;
  month_days[10] = 30;
  month_days[11] = 31;
  System.out.println("April has " + month_days[3] + " days.");
 }
}
```

Arrays

```
// An improvied version of the previous program.
class AutoArray {
  public static void main(String args[]) {
    int month_days[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,
     31 };
    System.out.println("April has " + month_days[3] + " days.");
  }
}
```

- The Java run-time system will check to be sure that all array indexes are in the correct range.

Arrays

- Multidimensional arrays
  - int twoD[][]=new int[4][5]

This image cannot currently be displayed.

End of Chapter 3

Questions?