Contents

[**Simple Class and Object** 1](#_Toc509946963)

|  |
| --- |
|  |
| Data Types |
|  |
| What is Variable? |
| * Named memory location   Example: byte a = 100; |
| What is? |
| * Each variable has type associated with it called as it is datatype. * Data type decides * The type and range of values that can be stored within variable (named memory location) * The set of operations that can be applied to the variable.   Example: byte a = 100;  There are two data types available in Java:   * Primitive Datatypes * Reference/Object Datatypes |
| What is Data type? |
| * Each variable has type associated with it called as it is datatype. * Data type decides * The type and range of values that can be stored within variable (named memory location) * The set of operations that can be applied to the variable.   Example: byte a = 100;  There are two data types available in Java:   * Primitive Datatypes * Reference/Object Datatypes |
| What is Primitive Datatypes? |
| Primitive (very basic) datatypes are predefined datatypes by the language.  Primitive variables store the actual values  There are 8 primitive types:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Datatype | Minimum value | Maximum value | Default value | Example | | Byte - 8-bit | -128 (-2^7) | 127 (2^7 -1) | 0 | byte a = 100;  byte b = -50; | | Short - 16-bit | -32,768 (-2^15) | 32,767 (2^15 -1) | 0 | short s = 10000; | | Int- 32-bit | (-2^31) | (2^31 -1) | 0 | int a = 100000;  int b = -200000 | | Long- 64-bit | (-2^63) | (-2^63 -1) | 0L | long a = 100000L,  long b = -200000L; | | Char- 16-bit | '\u0000' (or 0) | '\uffff' (or 65,535 inclusive) | Space | char letterA ='A' | | Float  - single-precision 32-bit |  |  | 0.0 | float f1 = 234.5f | | Double- double-precision 64-bit |  |  | 0.0 | double d1 = 123.4 | | Boolean-1 bit | true and false | true and false | False | boolean one = true | |
| What is Reference/Object Datatypes? |
| Reference variables store the addresses of the objects they refer to. Actual object stores in Heap memory.  Class, array comes under reference datatype.  Default value of any reference variable is null. |
| Literals and Escape Sequence |
|  |
| What is Literal ? |
| * A Literal is the source code representation of a fixed value. * literals are represented directly in your code without requiring computation. * Example : ~~int decimal =~~ 100; * byte, int, long, and short can be expressed in decimal(base 10), hexadecimal(base 16) or octal(base 8) number systems as well.   int decimal = 100;  int octal = 0144;  int hexa = 0x64;   * String literals Example:   ~~String s =~~ "Hello World"  ~~String s2 =~~ "two\nlines"   * String literals Example:   ~~char a =~~ 'A';   * String and char types of literals can contain any Unicode characters. For example:   char a = '\u0001';  String a = "\u0001";   * Java language supports few special escape sequences (\n, \r) for String and char literals as well.   Example : ~~String s2 =~~ "two\nlines" |
| What are Escape characters?  What are Escape sequences?  What are Escape codes? |
| * Escape characters (also called escape sequences or escape codes) in general are used to signal an alternative interpretation of a series of characters. * They are preceded by a backslash (\) * Java language supports few special escape sequences for String and char literals as well. |

|  |
| --- |
|  |
| **Simple Class and Object** |
|  |

|  |
| --- |
|  |
| What is Class? |
| A class is a blue print from which individual objects are created. A class can contain fields and methods to describe the behavior of an object. |
| What is Object? |
| * Object is a runtime entity and consist states and behaviours. * It’s state is stored in fields and behavior is shown via methods. * An object is an instance of a class.   Example:  A dog has states - color, name, breed as well as  behaviors – wagging the tail, barking, eating. |
| List the different ways for creating an Object for a class? |
| 1. Using the new keyword   It is the most common and regular way to create an object and actually very simple one also. By using this method we can call whichever constructor we want to call (no-arg constructor as well as parametrised).  Employee emp1 = new Employee();   1. Using Class.newInstance() method   We can also use the newInstance() method of the Class class to create objects, This newInstance() method calls the no-arg constructor to create the object.  We can create objects by newInstance() in any of the following way.   1. Employee emp2 = (Employee) Class.forName("org.programming.mitra.exercises.Employee")   .newInstance();   1. Employee emp2 = Employee.class.newInstance(); 2. Using newInstance() method of Constructor class   Similar to the newInstance() method of Class class, There is one newInstance() method in the java.lang.reflect.  Constructor class which we can use to create objects.  We can also call parameterized constructor, and private constructor by using this newInstance() method.  Constructor<Employee> constructor = Employee.class.getConstructor();  Employee emp3 = constructor.newInstance();   1. Using clone() method   Whenever we call clone() on any object JVM actually creates a new object for us and copy all content of the previous object into it.  Creating an object using clone method does not invoke any constructor.  To use clone() method on an object we need to implements Cloneable and define clone() method in it.  Employee emp4 = (Employee) emp3.clone();   1. Using deserialization   Object deserialization is nothing but creating an object from its serialized form.  ObjectInputStream inStream = new ObjectInputStream(anInputStream );  MyObject object = (MyObject) inStream.readObject(); |
| List the three steps for creating an Object for a class? |
| import java.util.Date;  class DateApp {  public static void main (String args[]) {  Date today = new Date();  System.out.println(today);  }  }  An Object is first declared, then instantiated and then it is initialized.   * Declaration: A variable declaration with a variable name with an object type.   Date today;  Or  Date today ~~= new Date();~~   * Instantiation: The new operator instantiates a new object by allocating memory for it.   new requires a single argument: a constructor method for the object to be created. The constructor method is responsible for initializing the new object.  ~~Date today =~~ new ~~Date();~~   * Initialization: The 'new' keyword is followed by a call to a constructor. This call initializes the new object.   ~~Date today = new~~ Date(); |
| What is Singleton class? |
| Singleton class control object creation, limiting the number to one but allowing the flexibility to create more objects if the situation changes. |

|  |
| --- |
| Java Overview |
| Variables - (Static,Instance,Local) |
|  |
| What is Variable ? |
| * A variable is a named memory location to store values that can be changed during program execution. * Each variable has type associated with it, called as datatype. Data type decides the type and range of values that can be stored within variable (named memory location) and the set of operations that can be applied to the variable. |
| How many types of Variable we have in Java? |
| There are three kinds of variables in Java:   * Local variables * Instance variables * Class/Static variables |
| What is a Local Variable? |
| Variables defined inside methods, constructors or blocks are called local variables.   * Local variables are created when the method, constructor or block is entered and the variable will be destroyed once it exits the method, constructor, or block. * Access modifiers cannot be used for local variables. * Local variables are visible only within the declared method, constructor or block. * Local variables are implemented at stack level internally. * There is no default value for local variables, so local variables should be declared and an initial value should be assigned before the first use. |
| What is Instance Variable? |
| Instance variables are variables within a class but outside any method. These variables are instantiated when the class is loaded.   * Instance variables are declared in a class, but outside a method, constructor or any block. * When a space is allocated for an object in the heap, a slot for each instance variable value is created. * Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed. * Instance variables hold values that must be referenced by more than one method, constructor or block, or essential parts of an object's state that must be present throughout the class. * Instance variables can be declared in class level before or after use. * Access modifiers can be given for instance variables. * The instance variables are visible for all methods, constructors and block in the class. Normally, it is recommended to make these variables private (access level). However, visibility for subclasses can be given for these variables with the use of access modifiers. * Instance variables have default values.   For numbers, the default value is 0,  For Booleans it is false  For object references it is null.   * Values can be assigned during the declaration or within the constructor. * Instance variables can be accessed directly by calling the variable name inside the class * However, within static methods (when instance variables are given accessibility), they should be called using the fully qualified name.   ObjectReference.VariableName. |
| What is a Class Variable?  What is a Static Variable? |
| These are variables declared with in a class, outside any method, with the static keyword.   * Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block. * There would only be one copy of each class variable per class, regardless of how many objects are created from it. * ~~Static variables are rarely used other than being declared as constants. Constants are variables that are declared as public/private, final, and static. Constant variables never change from their initial value.~~ * Static variables are stored in the static memory. * Static variables are created when the program starts and destroyed when the program stops. * Visibility is similar to instance variables. However, most static variables are declared public since they must be available for users of the class. * Default values are same as instance variables.   For numbers, the default value is 0  For Booleans, it is false  For object references, it is null.  Values can be assigned during the declaration or within the constructor.  Additionally, values can be assigned in special static initializer blocks.   * If the static variables are not public and final, the naming syntax is the same as instance and local variables. * Static variables can be accessed by calling with the class name ClassName.VariableName. * ~~When declaring class variables as public static final, then variable names (constants) are all in upper case.~~ |
| How variables stores? |
| * Static variables are stored in the static memory / Method Area.   Contains default values too.   * Object gets memory in Heap and object reference in Stack.   Object reference contains default value(null).   * Since Object gets memory in Heap , so does it’s object variable (instance variable). Contains default values too. * Local variables stores in Stack.   Contains No default values. |