**2 Java Programming Fundaments**

* 1. **Java Data Types :-**  A [**data type**](http://pc.net/glossary/definition/datatype) is a classification of data, which can store a specific type of information. Data types are primarily used in computer programming, in which variables are created to store data. Each variable is assigned a data type that determines what type of data the variable may contain.

The term "data type" and "**primitive data type**" are often used interchangeably. Primitive data types are predefined types of data, which are supported by the programming language. For example, [integer](http://pc.net/glossary/definition/integer), [character](http://pc.net/glossary/definition/character), and [string](http://pc.net/glossary/definition/string) are all primitive data types. Programmers can use these data types when creating variables in their programs. For example, a programmer may create a variable called "lastname" and define it as a string data type. The variable will then store data as a string of characters.

**Non-primitive data types** are not defined by the programming language, but are instead created by the programmer. They are sometimes called "reference variables," or "object references," since they reference a [memory](http://pc.net/glossary/definition/memory) location, which stores the data. In the [Java](http://pc.net/glossary/definition/java) programming language, non-primitive data types are simply called "objects" because they are created, rather than predefined. While an object may contain any type of data, the information referenced by the object may still be stored as a primitive data type.

This shows that there exists a field named 'pedal' that holds a data as a numerical value '1'. The values contained by the variables determines its data type and to perform the operations on it.  
**There are seven more primitive data types which are supported by Java language programming in addition to int.** A primitive data type is a data type which is predefined in Java. Following are the eight

**primitive data types:**

**int**It is a 32-bit signed two's complement integer data type. It ranges from -2,147,483,648 to 2,147,483,647. This data type is used for integer values. However for wider range of values use **long**.

**byte**  
The byte data type is an 8-bit signed two's complement integer. It ranges from -128 to127 (inclusive). We can save memory in large arrays using byte. We can also use byte instead of int  to increase the limit of the code.

**short**  
The short data type is a 16-bit signed two's complement integer. It ranges from -32,768 to 32,767. **short** is used to save memory in large arrays.

**long**  
The long data type is a 64-bit signed two's complement integer. It ranges from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. Use this data type with larger range of values.

**float**  
The float data type is a single-precision 32-bit IEEE 754 floating point. It ranges from 1.40129846432481707e-45 to 3.40282346638528860e+38 (positive or negative). Use a float (instead of double) to save memory in large arrays. We do not use this data type for the exact values such as currency. For that we have to use java.math.BigDecimal class.

**double**This data type is a double-precision 64-bit IEEE 754 floating point. It ranges from 4.94065645841246544e-324d to 1.79769313486231570e+308d (positive or negative). This data type is generally the default choice for decimal values.

**boolean**  
The boolean data type is 1-bit  and has only two values: **true** and **false**. We use this data type for conditional statements. true and false are not the same as True and False. They are defined constants of the language.

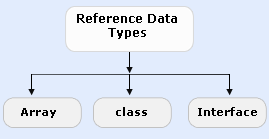
**char**  
The char data type is a single 16-bit, unsigned Unicode character. It ranges from 0 to 65,535. They are not same as ints, shorts etc.

The following table shows the default values for the data types:

|  |  |  |
| --- | --- | --- |
| **Keyword** | **Description** | **Size/Format** |
| byte | Byte-length integer | 8-bit two's complement |
| short | Short integer | 16-bit two's complement |
| int | Integer | 32-bit two's complement |
| long | Long integer | 64-bit two's complement |
| float | Single-precision floating point | 32-bit IEEE |
| double | Double-precision floating point | 64-bit IEEE |
| char | A single character | 16-bit Unicode character |
| boolean | A boolean value (true or false) | true or false |

When we declare a field it is not always essential that we initialize it too. The compiler sets a default value to the fields which are not initialized which might be zero or null. However this is not recommended.

* **Reference Data Types (Non-primitive data types)**Lets have a discussion about **Reference Data Types** in brief



In Java a **reference data type** is a variable that can contain the reference or an address of dynamically created object. These type of data type are not predefined like **primitive** data type. The reference data types are **arrays**, **classes** and **interfaces** that are made and handle according to a programmer in a java program  which can hold the three kind of values as:

|  |
| --- |
| **array type //** Points to an array instance  **class type //** Points to an object or a class instance  **interface type** // Points to an object and a method, which is implemented to the corresponding interface |

**class type:**

As you know that Java is an object-oriented programming language where an object is a  variable, associated with methods that is described by a class. The name of a class is treated  as a **type** in a java program, so that you can declare a variable of an object-type, and a method which can be called using that object- type variable.

Whenever a variable is created, a reference to an object is also created using the name of a class for its type i.e. that variable can contain either **null** or a **reference** to an object of that class. It is not allowed to contain any other kinds of values. Such type is called **reference types** in Java**.** The object becomes an **instance** when the memory is allocated to that object using **new** keyword. In addition, **array types** are **reference types** because these are treated as objects in Java. For example:

|  |
| --- |
| class Fruit {  fColor(){....}  fSize(){....}  };Fruit mango; Fruit banana; ... |

In the given example the **Fruit** is a **class** that has the reference variables as **mango & banana** through which we can call  
 the behaviors associated with that class as **mango.fColor();** within the main method of the super class.

**Array Type:**

|  |
| --- |
| An array is a special kind of **object** that contains values called **elements**. The java array enables the user to store the values of the same type in **contiguous** memory allocations. The elements in an array are identified by an **integer index** which initially starts from **0** and ends with**one less than number** of elements available in the array. All elements of an array must  contain the same type of value i.e. if an array is a type of integer then all the elements must be of integer type.  It is a **reference data type** because the class named as **Array**  implicitly extends **java.lang.Object**.The syntax of declaring  the array is shown as: |

or example:

|  |
| --- |
| **int [] a = new int [10]; String [] b = {"reference","data", "type"};** |

In the first statement, an array variable **"a"** is declared of  integer data type that holds the memory spaces according to the size of int. The index of the array starts from **a[0]** and ends with **a[9]**. Thus, the integer value can be assigned for each or a particular **index position** of the array.   
  
In the second statement,  the array **"b"** is declared of string data type that has the enough memory spaces to directly holds the three string values.  Thus each value is assigned for each **index** position of the array.

For more details about the arrays, click to the following link

**Interface Type:-**

 Java provides an another kind of **reference data type** or a mechanism  to support  **multiple inheritance** feature called an **interface.** The name of an interface can be used to specify the type of a reference. A value is not allowed to be assign to a variable declared using an interface type until the **object** implements the specified **interface**.

When a class declaration  implements an interface, that class inherits all of the variables and methods declared in that interface. So the  implementations for all of the methods declared in the interface must be provided by that class. For example, Java provides an interface called **ActionListener** whose method named **actionPerformed()** is used to handle the different kind of event . Java also provides a class called **Thread** that implements **Runnable** interface.   
Thus the following assignment can be allowed:

|  |
| --- |
| **Runnable r; r = new Thread();** |

.