# **Data Types**

**What are variable types?**

Variable types can be any data type that java supports, which includes the eight primitive data types, the name of a class or interface and an array.

**Name the eight primitive Java types.**

The eight primitive types are byte, char, short, int, long, float, double, and boolean.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Type** | **Bits** | **Bytes** | **Min** | **Max** | **Default** |
| **Byte** | 8 | 1 | -29 | 28-01-2001 | 0 |
| **Short** | 16 | 2 | -217 | 216-1-1 | 0 |
| **Int** | 32 | 4 | -233 | 232-1-1 | 0 |
| **Long** | 64 | 8 | -265 | 264-1-1 | 0 |
| **Float** | 32 | 4 | >NA | NA | 0.0f |
| **Double** | 64 | 8 | NA | NA | 0.0d |
| **Boolean** | 1 | NA | NA | NA | FALSE |
| **Char** | 16 | NA | NA | NA | '' |

**What is casting?**

There are two types of casting, casting between primitive numeric types and casting between object references. Casting between numeric types is used to convert larger values, such as double values, to smaller values, such as byte values. Casting between object references is used to refer to an object by a compatible class, interface, or array type reference.

**What is primitive casting in java?**

Primitive Casting is used to convert primitive values from one data type to another. For example, an int value can be assigned to a float data type, or a double value can be assigned to an int data type. Casting can be either implicit or explicit.

**Implicit Casting:** In implicit casting the conversion happens automatically, without writing specific code to do the conversion. Implicit casting happens when you convert or assign a smaller value, like a byte, to a larger data type such as an int.

**Explicit Casting:** In explicit casting code has to be specifically written to perform the conversion from one primitive type to another. Explicit casting is done by using the syntax (data\_type) where data\_type is the data type that the cast is being applied to. Explicit casting happens when you convert or assign a larger value to a smaller data type.

**package** intquestions;

**class** testclass

{

**public** **static** **void** main(String[] args)

{

**byte** b = 2;

**int** i = b; //Implicit casting

**int** j = 3;

//byte k = j;//Throws error

**byte** k = (**byte**)j; //Explicit casting

}

}

**Are arrays primitive data types?**

In Java, Arrays are objects.

**What are the default values of primitive data types?**

The default values are given by the following example.

**package** intquestions;

**public** **class** testclass

{

**byte** b;

**short** s;

**int** i;

**long** l;

**float** f;

**double** d;

**boolean** bo;

**char** ch;

**public** **static** **void** main(String[] args) {

testclass tc = **new** testclass();

System.***out***.println("byte default value is "+tc.b);

System.***out***.println("short default value is "+tc.s);

System.***out***.println("int default value is "+tc.i);

System.***out***.println("long default value is "+tc.l);

System.***out***.println("float default value is "+tc.f);

System.***out***.println("double default value is "+tc.d);

System.***out***.println("boolean default value is "+tc.bo);

System.***out***.println("char default value is "+tc.ch);

}

}

Result:

byte default value is 0

short default value is 0

int default value is 0

long default value is 0

float default value is 0.0

double default value is 0.0

boolean default value is false

char default value is

**What are Primitive Literals?**

Primitive Literals are the code representation of values of primitive data types. For example 'a' is a char literal, 100 is an int literal, 'false' is a boolean literal and 2345.456 is a double literal.

**How is rounding performed under integer division?**

The fractional part of the result is truncated. This is known as rounding toward zero.

Example:

**public** **class** JavaExamples {

**static** **int** *a* = 5;

**static** **int** *b* = 2;

**public** **static** **void** main(String[] args) {

System.***out***.println("a/b is "+5/2); //Rounding is performed by truncating the fractional part.

}

}

Result: 2

**To what value is a variable of the boolean type automatically initialized?**

The default value of the boolean type is false.

**What is type conversion in java?**

Assigning a value of one type to variable of other type is called type conversion.

Example:

int a =10;

long b=a;

There are two types of conversion in java:

1) Widening conversion

2) Narrowing conversion

**Explain about Automatic type conversion in java?**

Java automatic type conversion is done if the following conditions are met :

1. When two types are compatible

Ex: int, float

int can be assigned directly to float variable.

1. Destination type is larger than source type.

Ex: int, long

Int can be assigned directly to long. Automatic type conversion takes place if int is assigned to long because long is larger datatype than int. Widening Conversion comes under Automatic type conversion.

**Explain about narrowing conversion in java?**

When destination type is smaller than source type we use narrowing conversion mechanism in java. Narrowing conversion has to be done manually if destination type is smaller than source type.

To do narrowing conversion we use cast. Cast is nothing but explicit type conversion.

Example:

long a;

byte b;

b=(byte)a;

Note: casting to be done only on valid types otherwise classcastexception will be thrown.

**Can you compare a boolean with an int variable in Java?**

No. you will get compilation error.

Example:

**public** **class** JavaExamples {

**static** **int** *a* = 5;

**static** **boolean** *b* = **false**;

**public** **static** **void** main(String[] args) {

**if**(*a* == *b*) { //Error: The operator == is undefined for the argument types int, boolean

System.***out***.println("Comparing boolean with int");

}

}

}

**Difference between double and float variables in Java.**

In java, float takes 4 bytes in memory while Double takes 8 bytes in memory. Float is single precision floating point decimal number while Double is double precision decimal number.

**What is the output of the following?**

System.out.println(**1.0**/**0**);

Most of us may expect ArithmeticException, however, in this case, there will be no exception instead it prints **Infinity**.

1.0 is a double literal and double datatype supports infinity.

Example:

**package** intquestions;

**public** **class** testclass {

**public** **static** **void** main(String[] args) {

System.***out***.println("value is "+(1.0/0));

}

}

Result:

value is Infinity