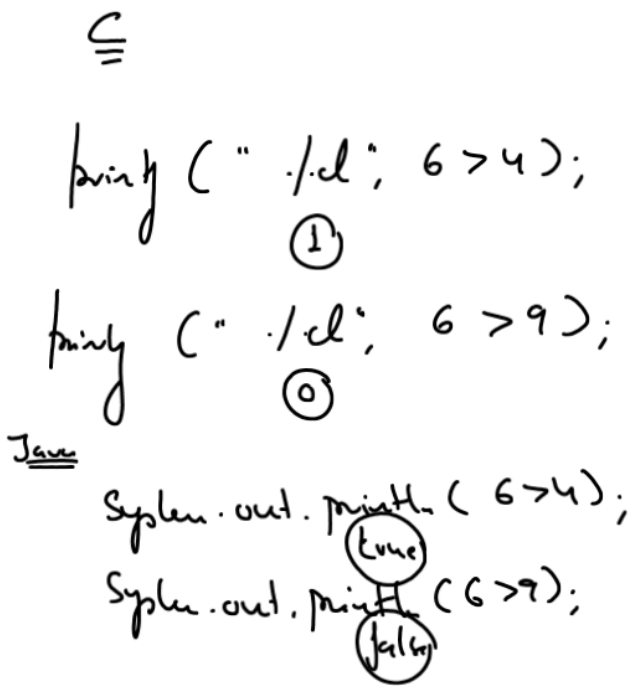
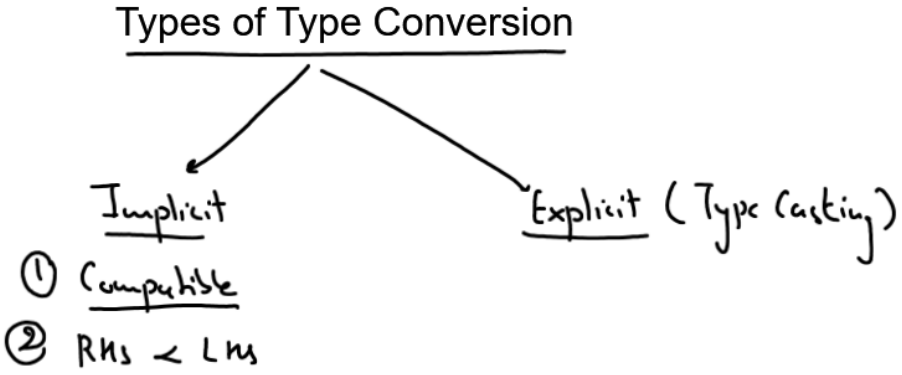
Data Type Name	Size (In Bytes)	Range
① byte	1B	-128 to 127
② short	2B	-32768 to 32767
③ int	4B	-2147483648 To 2147483647
④ long	8B	-9,223,372,036,854,775,808 To 9,223,372,036,854,775,807

Data Type	Size	Range
⑤ char	2B	0 — 65535
⑥ boolean	1B (8bit) OR 1 bit	true, false
⑦ float	4B	-3.4×10^{38} to 3.4×10^{38}
⑧ double	8B	-1.7×10^{308} to 1.7×10^{308}



Type Conversion

Type conversion is a concept which is applied by Java compiler whenever be use assignment operator between variables of different datatypes.



```
int x;
x = 'A';
```

```
int x;
x = true;
```

Error: Incompatible Types: Cannot convert boolean to int

```
double x;
x = 10;
```

```
int x;
x = 1.7;
```

```
int x;
x = 0.0;
```

Error: Possible lossy conversion from double to int

```
x = (int)1.7;
```

①

Types of Type Conversion
==========
In Java, type conversion is broadly classified to be of two types:

1. Implicit Conversion (Done by compiler)
2. Explicit Conversion (To be done by programmer)

Rules of Implicit Conversion
==========
1. Values must be compatible i.e. there must be some way in Java to convert RHS to LHS.

For example:
int x;
x='A';

The above code will run because Java will convert character constant 'A' to its unicode equivalent 65 because character is **compatible** with integer.

But
int x;
x=true; //The above code will not even compile because the Java compiler **has no way** of converting boolean to int. So the error will be **incompatible types: can't convert boolean to int.**

2. The value on RHS of assignment operator must be smaller in terms of **range** as compared to the variable on LHS.

For example:
double x;
x=10;
The above code will compile and run because 10 is an integer which is a **smaller type** as compared to the data type of x which is double.
But
int x;
x=10.5;

The above code will not even compile because 10.5 is **double** which is of **higher type** as compared to the datatype of x which is int and the error message will be **possible lossy conversion from double to int**
If we want the above code to run then we will have to use **explicit converison** as shown below.
int x;
x=(int)10.5;
Now, the code will compile and run but x will store 10.