WELCOME TO JAVA CLASS ON TYPE CASTING

Type Casting

Java is a **strongly-typed** programming language. This means we must declare data types before you can make use of the variables or methods.

```
int x =5;
```

```
double valueOfPie = 3.141;
public static int getSum(int x, int y){
int myNumber = 7;
return sum;
}
```

byte a = 126;
byte b = 126;

int sum = a + b; (252)

Cast = to give shape.

Type casting is used to convert an object or variable of one type into another.
Or, Assigning a value of one type to a variable of another type is known as Type Casting.

There are three situations in which conversion of primitive type takes place

- Assignment:
  - double x;
  - int y = 50;
  - x = y;
- Arithmetic Promotion:
  - int count =7; double sum = 24;
  - double average = sum / count; //count is also converted to double
  - byte b = 5;
- Method call:
  - float f = 1.234f
  - double d = Math.cos(f); //method cos() expects double, hence f is converted to double

Variables are named containers in the memory and they hold values.

Smaller to bigger data types: `byte < short < int < long < float < double`

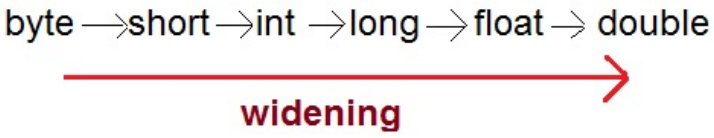
2 kinds of **type casting:**

A. **Widening or automatic or Implicit type casting**:

the two types are compatible

the target type is larger (wider) than the source type

smaller to larger type



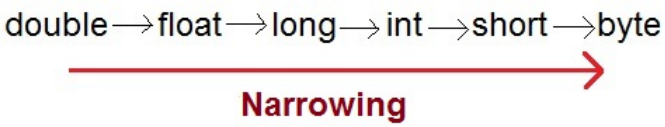
Coding example:

```
byte myByte = 12;

int myInt = myByte;

System.out.println(myInt);
```

B. Narrowing or Explicit type casting: When you are assigning a larger type value to a variable of smaller type, then you need to perform explicit type casting.



Coding example:

```
int anotherInt = 120;
byte anotherByte = (int) anotherInt;
System.out.println(anotherByte);
```

Automatic Type promotion:

All ***byte*** and ***short*** values are promoted to ***int***.
If one operand is a ***long***, the whole expression is promoted to ***long***.
If one operand is a ***float***, the entire expression is promoted to ***float***.1.232f x 23
If any of the operands is ***double***, the result is ***double***.

```
byte b = 4;
float f = 5.5f;
float result = (f * b);
System.out.println("f * b = " + result);
```

5.49152356E8

Advantage of Autoboxing and Unboxing:

Autoboxing: Converting a primitive value into an object of the corresponding wrapper class is called autoboxing.

```
public static void main(String args[]){
  int a = 50; // a is a primitive data for int variable and holds the value 50
    Integer a2=new Integer(a);  // a2 has the value of 50 through a (int a = 50);
//Boxing, Integer represents the Integer (Wrapper Class)

     Integer a3=5;//Boxing

     System.out.println(a2+" "+a3);
}
```

Unboxing: Converting an object of a wrapper type to its corresponding primitive value is called unboxing.

```
public static void main(String args[]){
   Integer i =new Integer(50);
      int a = i;         // a represents primitive data type - int type
      System.out.println(a);
 }
}
```