**Java-8 – Auto-boxing, Auto-unboxing and Type parameter**

* **Primitive Type Functional Interfaces:**

To understand about the primitive functional interface, we need to know the following topics.

Autoboxing:

Conversion of primitive to object type automatically is called autoboxing.

Compiler is responsible for this automatic conversion.

This feature got introduced in version 1.5

Example:

Integer I = 10;

Converted to, Integer I = Integer.valueOf(10);

Autounboxing:

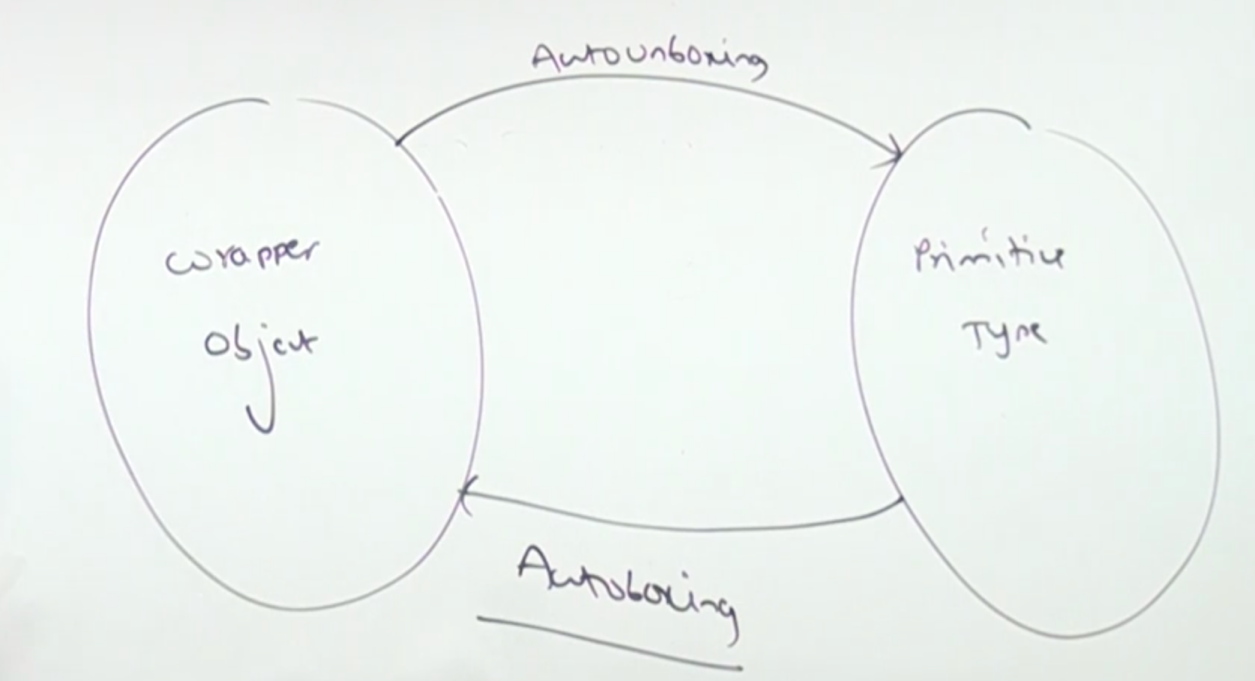
Conversion of wrapper object to primitive is called autounboxing.

Example:

Integer I = new Integer(10);

int x = I;

Converted to, int x = I.intValue();



Generics – Type Parameter:

ArrayList<Integer> l =new ArrayList<Integer>();

Integer – is called type parameter.

It will accept only object type not primitive.

* **Need of primate type functional interfaces:**

Predicate will only take Object as the input type.

Let us say we have to write a Predicate to check the given integer is even or not. Below is the code to check.

int x = {0, 5, 10, 15, 20, 25};

Predicate<Integer> p = I -> I % 2 == 0;

for(int x1: x){

if(p.test(x1)){ // Line-1

System.out.println(x1);

}

}

The above lines of code performance wise slow because of the following reasons.

1. At line-1 it has to do the autoboxing as the Predicate will accept only Object type as an input.
2. At the time of division check again it has to perform autounboxing as we can’t do the division directly with object type.

For the above input, 6 times autoboxing and 6 times auto unboxing will occur which will decrease the performance.

Note: Normal functional interfaces are not suitable for primitives.

Function<Integer, Integer> f = I -> I \* I;

System.out.println(f.apply(10)); // Line-1

The same analogy applies here as well, at Line-1 and multiplication code block

* **Primitive type functional interfaces for Predicate:**

interface IntPredicate{

public boolean test(int i);

}

class Test{

public static void main(String[] args){

int[] x = {0, 10, 15, 20, 25};

IntPredicate p = I -> I % 2 == 0;

for(int x1: x){

if(p.test(x1)){

System.out.println(x1);

}

}

}

}

* **Similar primitive predicates:**

interface LongPredicate{

public boolean test(long l);

}

interface DoublePredicate{

public boolean test(double d);

}

* **This document contains contents of 66, 67 & 68**