# COLLECTIONS

# ARRAYS

**Array** – is object that stores collection of values

Array can store 2 types of data:

* **A collection of primitive values** –
* **A collection of objects** – in fact it’s heap memory

Creating array involves 3 steps:

* **declaration** –

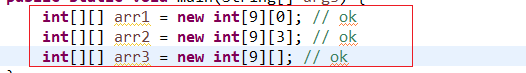
int[] arr1; - one dimensional array

int[][] arr2; - multidimensional array

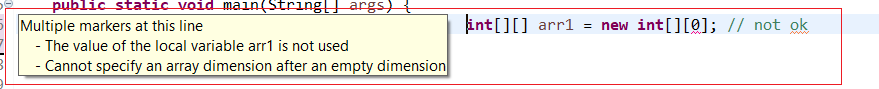
int[] arr2[]; - multidimensional array (another form)

* **allocation**

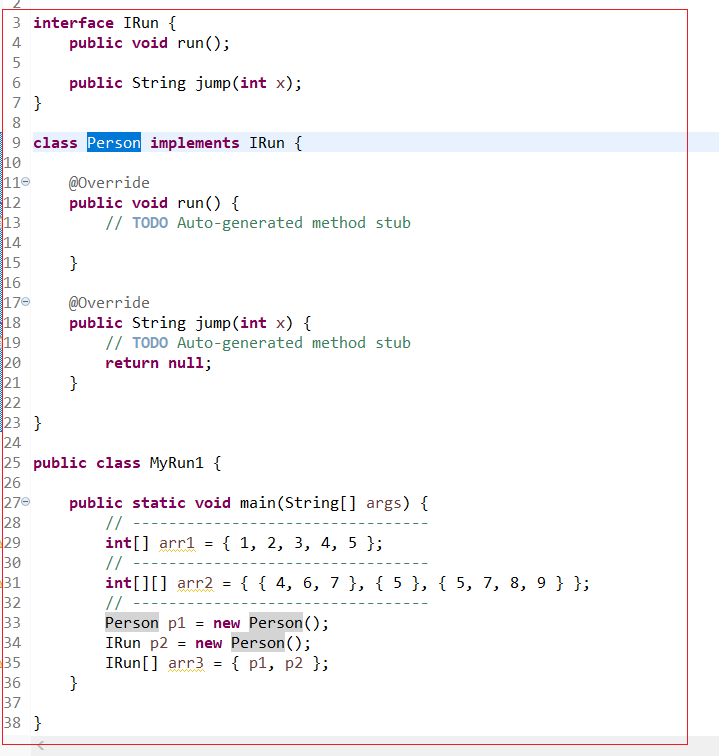
*example of right declaration:*



*example of not right declaration:*



* **initialization**



# ARRAYLIST

Array can’t change size once created. ArrayList does it automatically

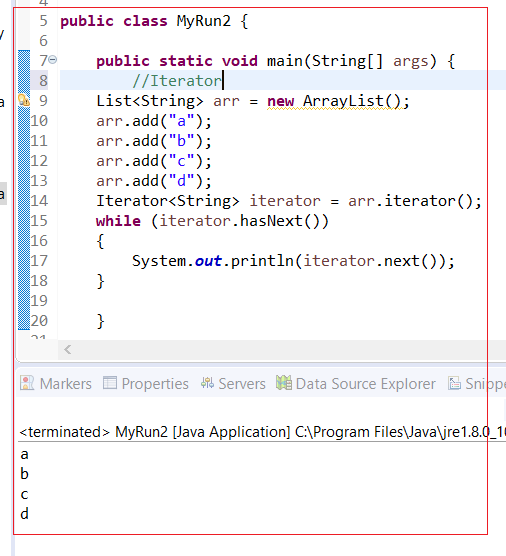
* Arraylist cannot contain primitives



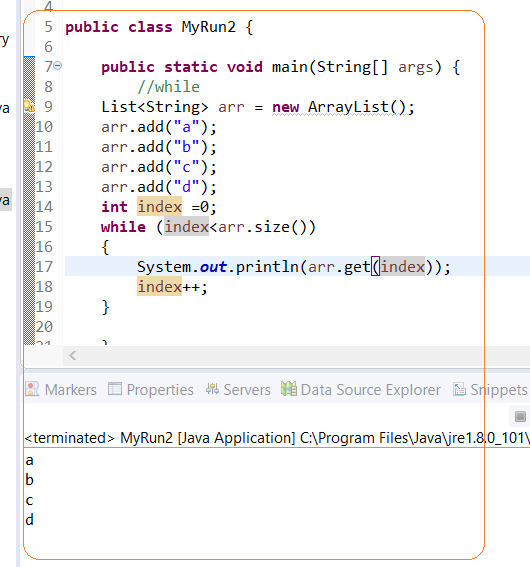
There are 5 ways to iterate through loop:

* **For Loop** -
* **ForEach** -
* **Iterator** -
* **While Loop** -
* **Collections’s stream() util (Java8)** –

**ITERATOR**

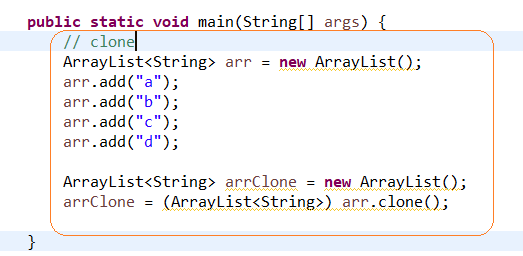


**WHILE**



**CLONE**

**Cloning –** it does not clone Object. It clones only reference of elements of object



# GENERICS

## OIVERVIEW

* Before java 5 you had to write code like this



and hope that programmers remember that you wanted only String

* In Java 5 you could parametrized types



* when Java 7 came you could use shorten form

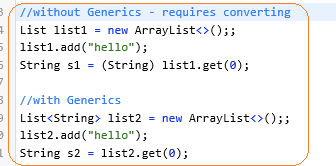


**Diamond operator** - The shorten form <> is called diamond

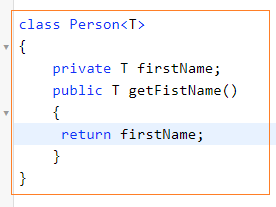
## MAIN

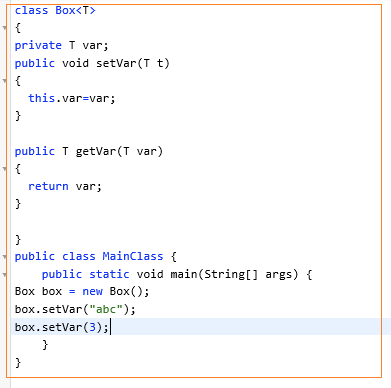
**Generics** - provide compile time checking and removing risk of [**ClassCastException**].

* It allows to avoid run time error **[ClassCastException**] and if there mistake throws error at compile step
* We don’t need to make additional casting



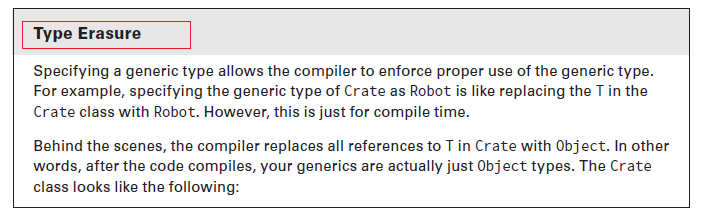
* Help to reuse code

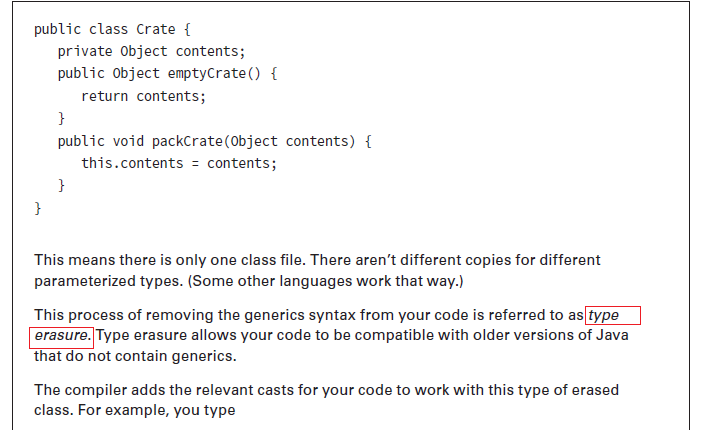




## ERAUSERE

**Type Erasure** – when compiler compiles your code it replaces all references (like <T>) to Object class with necessary casting. It’s done to be compatible with early versions of Java when there were not generics





## GENERIC BOUNDS

|  |  |  |
| --- | --- | --- |
| TYPE OF BOUND | SYNTAX | EXAMPLE |
| **UNBOUNDED** | **<?>** | List<?> list = new ArrayList<String>(); |
| **UPPER BOUNDED** | **<? extends type>** | List<? extends Number> list = new ArrayList<Integer>(); |
| **LOWER BOUNDED** | **<? Super type>** | List<? super Exception> list = new ArrayList<Object>(); |

UPPER BOUNDED:



# COLLECTIONS

**Collections interfaces:**

* List
* Set
* Map
* Queue