Generalization is the process of taking out common properties and functionalities from two or more classes and combining them together into another class which acts as the parent class of those classes or what we may say the **generalized** class of those specialized classes.

```
Object obj = new String(“Hi”);

/*-------------------------------------------------------------------------*/

public class GenericDemo<T>

//Generic array

T data[] = (T[]) new Object[3];  /*(T[]) -> typecasted. It can
store any type of object but before using it we should
mention type.*/


/* I cannot directly use this because main is static type  and
this is not static. I should create an object of generic demo
and use it.*/

Public static void main(String[] args) {

   GenericDemo<String> gd=new GenericDemo(); /* type is
String. So, once I mention string here it becomes a String type
instead. Now T becomes String type.*/

/*Store some value. Remember it can store any value type
but here I mention it String type.*/

gd.data[0]=”hi”;

gd.data[1]=”bye”;
gd.data[2]=10; /*error -> why? Integer cannot be converted
into a String. So it just become a compiler error.*/

gd.data[2]=new Integer(10); /*error -> here I will create an
object of type integer. So, I cannot assign an integer because
I mention earlier that I’m going to store a string. */


String str=gd.data[0]; /*I don’t have to do typecasting. So,
the befit of Generic is that we can have a generalized array,
but while using we can mention the type as parameter and
another benefit is at compile time only it will check that we
are not storing anything invalid. We are strong same type of
objects.*/


}
```