02/07/2024

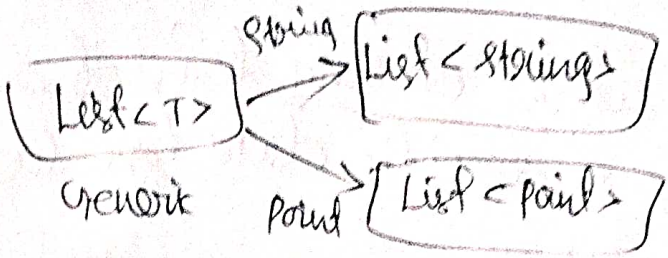
LinkedList:

* It is an implementation of the List and Deque interfaces
* Internally, it is an implemented using Doubly linked list Data structure.
* It supports duplicate elements.
* It stores or maintains its elements in Insertion order.
* We can add any number of null elements.
* It is not synchronized that means it is not Thread safe.
* We can create a synchronized LinkedList using collections.synchronizedList() method.
* In java applications, we can use it as a list, stack or queue.
* We can use ListIterator to iterate LinkedList elements.

Generics with Syntax and Example

* Generics is a mechanism by which a single piece of code can manipulate many different data types without explicitly having a separate entity for each data type.
* The Java Generics allows us to create a single class, interface, and method that can be used with different types of data (objects)

* Generics were used to check the type compatibility at the compile time and hence removing the chances of occuring class cast exception at runtime.
* Generics, as the name suggests, is a general way of defining methods, definitions, and collections so that they are not dependent on the data type supplied to them.
* Generics allows programmer to create parameterized types instances of such types can be created by passing reference types.



* They can function the same no matter what data type they use.
* Collections such as ArrayList use generics extensively.
* Generics are generally declared in the "<>" brackets.

Syntax:

```
class Class_name <T1, T2, T3... Tn>
{ // Generic Type or Parameterized type
}
```

where T1, T2, T3.., Tn (T stands for Type) enclosed within angle bracket (<>) are called type parameters and class 'Class_name' is called generic type or parameterized type.

Generic Class

* A class that is declared generic type is called generic class in Java. It is type-safe and can't upon any data type.
* A generic class is also called "parameterized type" because it uses a parameter that determines which data type it should work upon.
* Java generic class is designed to work upon objects. Hence, it cannot work with primitive data types.
* Generic class works on any kind of datatype we cannot specify a specific data type such as integer, String, etc.

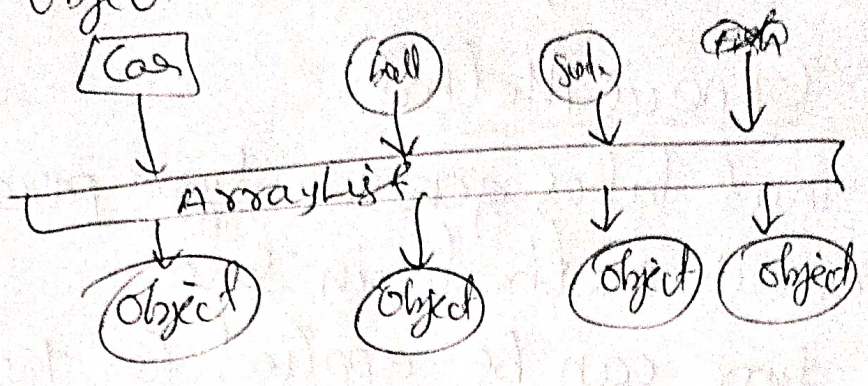
```
class MyClass<T>
  { // class code
  }
```

Ex:

```
public class Sender<T> {
    private T message;
    public setMessage(T message){
        this.message = message;
    }
    public sendMessage(){
        // logic to send message
    }
}
Sender<String> stringSender = new Sender<String>();
Sender<Employee> empSender = new Sender<Employee>();
```

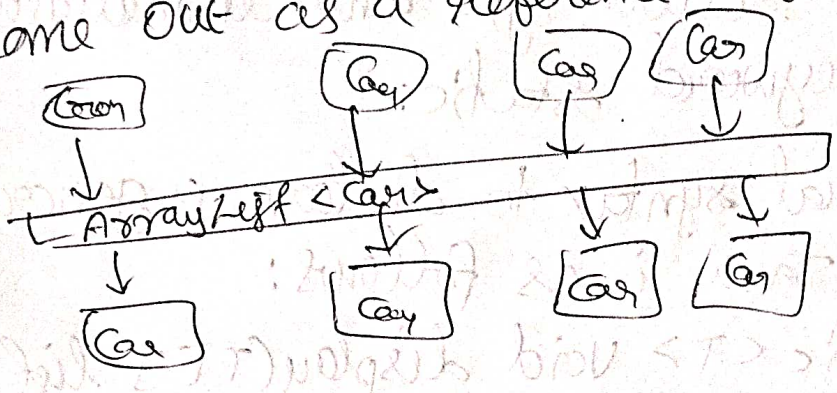
Without Generics:

Objects go in as a reference to Car, Football, Scooter, and Fish objects. and come out as a reference of type object.



With Generics:

Objects go In as a reference to only Car objects. and come out as a reference of type car.



Need of Generics:

* Errors are integral part of coding. Some errors occur at compile time and some errors occur at run time.
* Errors which occurs at compile time can be easily identified and can be removed. But, run time errors occurs when an application is running in real time. If they happen, they cause abrupt termination of an application.
* ClassCastException is also such an exception which happens only at run time.

* It occurs when data of one type cannot be casted to another type.
* You will never get a single clue about this exception during compilation.

Generic Method

* A method that takes generic type parameters is called generic method in Java.
* A generic type can be applied for the static method. We can define a generic method by putting generic type parameter <T> before the method return type and immediately after the keyword static.
*. The general syntax to declare a generic method in Java is as follows:

```
public static <T> void display(T[] list)
{
    // method code;
}
```