Wednesday, May 4, 2022 10:15 AM

```
  class Thread  implements Runnable
  {
        Runnable obj ;

        Thread(Runnable task)  { obj = task;  }

        Public void run()
        {
                If( obj != null)  {   obj.run(); }
                Else
                {
                }
        }

   }
```

____

In overriding method we can reduce exceptions in overridden method

Super	throws Exception
Sub	throws IOException

In overriding method we CANNOT increase exceptions in overridden method

Super	throws IOException
Sub	throws Exception

____

Collections ----------   Inbuilt implementation of Data Structures

java.util package .

Collection interface

Declare abstract methods for general activities on collections

Notebook --- collection of names for my birthday party--- list

General/Common actions that I can perform on collection of names --------

search a name is in the list or not
sort list alphabetically
modify element of list
add (append)   , **insert** in between
remove/delete element in the list
count
shuffle
Traverse --- visit each element of the list

Instead of writing names in the notebook , I want to store the names in the RAM .....

welcome to data structures -------

Array

Linked list, doubly linked list , circular linked list
Queue
Dequeue
Circular queue,
Stack
Tree
Graph
Hash table

____

Generics <DATATYPE>

Down casting rule
(subclass-type)Superclass-ref

Upcasting rule
Super-class-type = sub-class-object

____

Generics ------- to overcome the **ClassCastException**
People wanted the compiler to inform at compile time instead of crashing at run time

1. I want to have a class MyStack , it should be usable for any data type
2. But using Object[] is risky as it can have **mixture** of different data types- which may lead to **ClassCastException**.
3. Hence Generics use Placeholder and Actual type parameters to solve the problem.

84,93,113------- absent but connected

You will use generics rather than writing generic classes!!!!!

____

ready made implementation --------- QUICK , we don't have to spend time to implement the data structure
+
API is written by best coders --- space time complexities are taken into consideration
We get good efficient algorithms

____
_

Interface java.util. Collection

Interface List is a Collection
List can have INDEXED ACCESS (0 th element , 1st element ….)
Duplicate elements can be added to the list

Interface Set is a Collection
NO INDEXED ACCESS

Duplicate elements are not added in the set --- ONLY unique elements

---

Ex1 ----- Write a class TestList study.collections

ArrayList is a subclass of List

It is a growable array --- all elements are stored in consecutive locations and they can be accessed using index

ArrayList	Vector
Subclass of List	Subclass of List
Growable array	Growable array
Not thread safe	Thread safe

---

java.util Iterator is an interface

boolean hasNext ---------whether next element is present in the collection

E next() -------------if next element is present then next goes to that element and returns it.

remove() ----------- removes the **current** element

Iterator is used for TRAVERSING a collection

We get the object of the iterator subclass when we call an API **iterator()**

HW ---

1. Complete generic stack example
2. Type the TesList.java and TestList2.java and observe
3. Try different remove methods in TestList2 as discussed in class
4. Write a class BirthdayList

   add a property ArrayList<String> guests

   methods

   int howManyPeople() -----call the size API and return result

   void addAName(String name) --- call add API

   void removeAName(String name) ---- call remove API

   void showGuestList() ---- use Iterator and show names void clearList() ----use clear API

boolean haveIAddedThisName(String name) --- call contains API

Write a User class

main

menu

a. Show guest list
b. Remove a name
c. Add a name
d. Clear list
e. Is the name in the list
f. Total guests
g. Quit

____