Thursday, November 18, 2021 10:27 AM

Try block ----- we write the code that may have a problem at run time

Catch block ---- we write the code to handle the exception

1. Apply some logic
2. Give a proper USER FRIENDLY message
3. Print stack trace

Finally block ---- we write the code that MUST be executed

1. If no exception occur
2. If exception occurs and it is handled
3. If exception occurs and it is not handled (crash situation)

Possible combinations ----

try-catch
try -catch-catch -catch -catch (remember that base class catch must be in the end)
try-catch-catch-finally
try-finally

HW ---- Play with the TestFinally1 and 2 classes written during session!!!

Observe the conditions --

1. If no exception occur
2. If exception occurs and it is handled
3. If exception occurs and it is not handled (crash situation)

Exception Chaining ---------

```
Void  f1()
{
        Condition
                Throw new CheckedException();
}
```

```
Worker-------------- faces a  problem
        Supervisor
                Manager
                        Director
```

If worker faces a problem

a. Worker solves it himself (try - catch)
b. Worker cannot solve it
    a. Worker may propagate the problem to supervisor (throws)
        Supervisor has 2 options
        a. Solve it (try- catch)
        b. Escalate it / propagate it to Manager (throws)
            1. Manager has 2 option
                a) solve it (try - catch)

b) Escalate /propagate to Director (throws)

2. Director has 2 options   MAIN method
   a) Solve it (try - catch)
   b) CRASH (throws)

```
Exception
        |
        RunTimeException
                |
                NumberFormatException  (  UNCHECKED  )
```

HW --- Play with TestExceptionChaining code done in class !!!

QUESTION---

WHEN you are overriding a method that throws any exception
While overriding can you change the throws part in the subclass
Then can you make it broader(super class) or narrower(subclass)!!!

```
Class A
{
        Void f1() throws IOExeption
        {
        }
}


Class B  extends A
{
        @Override
         void f1()   throws Exception  ???  VALID HAI  ????
        {
        }

        OR
        @Override
         void f1()   throws FileNotFoundException  ???  VALID HAI  ????
        {
        }
        OR

        @Override
         void f1()   VALID HAI  ????
        {
        }
        OR
        @Override
         void f1()   throws NumberFormatException  ???  VALID HAI  ????
        {
        }

        OR
        @Override
```

```
                        void f1()   throws IOException  ???  VALID HAI  ????
                        {
                        }

                }
```
____

Interface ----- by default all methods are ABSTRACT !!!

____

Iwork
  work , takeOff (default impl -paid leave)

Class ME implements Iwork
 --------------------------------------It is getting default Impl of takeOff
 It must override work !!! It may override takeoff !!
____

Iplay
 Play , takeoff (default impl -- take rest)

Class ME implements Iplay
 It must override play()
 It may override takeoff() // it gets default takeoff if not implemented

____

```
Class ME implements Iwork ,Iplay   }}}  MULTIPLE INHERITANCE
{
            It must override work
            It must override play
            CLASH/CONFUSION/AMBIGUITY   ----- for takeoff  ????WHICH DEFAULT IMPL is it
            getting  ????
            It  MUST override takeoff   !!!!! // default advantage is GONE
}


ME obj = new ME();
obj.takeoff()  ;    //overridden takeoff will be called
```

____

HW --- Try Out the Iwork, Iplay example done in class .
____

Generics !!! I want to have a **stack of any data type** by writing a **single** class
 If Object[] is used then this can hold a mixture of all data types (CHIVDA) ---- this is not acceptable
 Because at run time we may get ClassCastException

 Solution to this problem is GENERICS !!!
  class MyGenericStack < T >

  T is a place holder of data holder !!!
____

Collections !!! API libraries that are implementing the data structures !!!

Java . Util PACKAGE

interface Collection
All methods for common operations on Data structures

Data Structures ---
Array , linked list , stack, queue, tree , graph(heap) , hashtable !!!

To hold collections of items in RAM !!!!
paper pen --- I create a list of grocery

1. Sugar
2. Tea
3. Maggi
4. Rice
5. Coffee

-- **Common Operations** that can be performed on collection of items
Insert element , append , remove, search , sort , traverse-each-element , replace/modify , count , clear , add alist to another list , isempty

---

Class **extends** class !!!
interface **extends** interface !!!
class **implements** interface !!
interface implements interface !!!! KABHI NAHI HOTA !!!!!!!!

List interface is a Collection ---
It uses index for accessing elements ---- INDEX BASED access
It can have duplicate elements

Set interface is a Collection -----
It DOES NOT use index for accessing elements
It cannot be duplicates !!

HW

```
Write a class ArrayListExample
        Main
        Switch case -
                Call populatelist
                Call showlist
                 call sum
                Call remove element

        Public static void showList(List<Integer"  list)
        {
                 show all elements in the list
        }

        Public static int sumAll(List<Integer>  list)
```

```
{
        Calculate sum of all integers in the list
}


Public static void populateList(List<Integer> list )
{
        Ask user whether to insert or append
           1.  Insert   ---- ask index and value   and call   add(index,value)
           2.  Append  --- call  add(value)
        Go on asking till user enters "no"
}

Public static void removeElement(List<Integer> list)
{
        Ask user whether to remove by index or by value
           1.  Index    remove(index)
           2.  Value   remove(value )
}
```
