we can use try-with-resources to close all your resources it keeps all your exception prone lines in single block.
try-with-resource is method defined in autoclosable interface

5/12/22

* Code which safely work with all data types is called generic programming
To implement this we could have used overloading ~~but in th~~ and overriding (polymorphism) but the code mainability hampers
So if we want to achieve both "type safety and maintainability"
We can use parametric polymorphism

* Parametric polymorphism
passing data type as a arguments is called parametric polymorphism

eg.
```
private static <T> T Select (int choice, T first, T second){
    if((choice %2) == 1)
        return first;
    return second;
}

p.s.v.m (String args[]) {
 String s = Select ( 23, "Monday", "Tuesday");
Double d = select (32, 43.2 , 34.5);
Interval si = Select (new Interval(2,30), new Interval(3,65));
}
```

here we can pass T as primitive types, String or ~~class~~ object of class (userdefine class)
T is considered as "java.lang.object" also called Type Eraser.

Interval is not java class it is created by us and we are passing it as parameter ∴ here along with values data type of values (arguments) is also being passed to "select()".

* Comparable Interface

We cannot compare object

We can only compare primitive datype using "less than" < "greater than" > operator

To compare objects we have to use "CompareTo()" but since whatever we passed in "< >" angular brackets is considered as java.lang object but compareTo() is not defined object class of lang package cuz Every object can have ~~its~~ ~~own~~ their own method of comparing it and this method should be defined its class eg. object of Student can be compared according to percentages as well as roll no.

That's why java provides us with CompareTo() is in string class and Double class and can only work on String & Double.

That's why java provides interface called Comparable which has only method CompareTo() which can be overrided by implementing Comparable<Type>

eg.

```
private static <T extends Comparable<T>> T select
(T first, T second) {
    if (first.CompareTo(second) >0)
        return first;
    return second;
```

```
class Interval implements Comparable<Interval> {
  // logic of interval
   public int compareTo (interval that) {
     // logic to compare
  }
```

* Autoboxing

Compiler does Autoboxing

autoboxing is conversion of primitive type into its corresponding wrapper class type by putting primitive type value into instance of its wrapper class

* Inner class

class inside class but inner class is non-static. this class can refer to non-static members of outer class

Classes can be made generic by

```
class classname <T> {
   T field1;
   T field2;
  public void method1 ( T param) {
      //logic
  }
  public T method2 ( T param) {
     //logic
  }

  p.s.v.m () {
classname<T> obj = new classname<T> ();
}
```

* Varience

Mango is fruit → True

but basket of mango ~~Can~~ is basket of fruit → False

because whatever can be done with fruit can be done with mango like eat(), cut() etc but if we add orange in basket of fruit it will still be a basket of fruit but if we add orange in basket of mango it will be no longer a basket of mango.

that is invarience when a collection of a specific type is required only a collection with this type can be used.

If we are giving Dee a ~~mango~~ ~~at~~ basket of mango then she can only count() or see() those can mango and cannot add any mango.

Covarient means ~~can add~~ ability to substitute a collection of subtype when a more general type is needed

If we ~~add~~ give dee ~~an~~ basket of ~~mago~~ mango the mango can be added or if we give her basket of fruits then also she can add mangos to it and it will still be basket of fruits this is contravarient

The ability to substitute a collection of more general type when a collection of more specific type is needed.

detination in techical terms:

1) Invarient :- substitution is not possible even if it is invarient

department <employee>
↑ not possible
department < salesperson

2) Covariente :- Substitution of subtype is possible but cannot call methods which ~~appears as~~ are appearing in subtype and has parameter passed as subtype eg. void ~~pop~~ push (V item)

3) Contravarient :- Substitution of superlype is possible ~~in sub~~ with subtype but cannot call those method appearing in subtype which have Return type same as subtype eg V pop();

Covarience :- eg. department < ? extends Employee>
Syntax
↑ possible but cannot call method having parameter passed as ~~subtype~~ salesperson

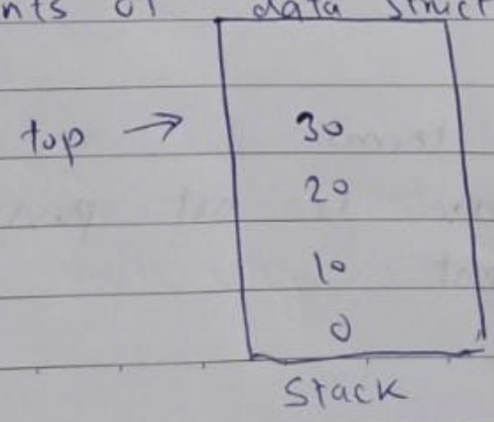
department <salesperson>

Contravient :- department < Salesperson>
↑ possible but cannot call method having returntype salesperson.

department < Employee>

* Iterations :-

Used when we have has a relationship. we are able to visit things without changing contents of data structure



current is iterator

Iterable is interface in java which has method Iterator() we can always override this method for iterations. which returns instance of iterator interface

* Anonymous class

~~class written inside method is called anonymous class.~~ A class without name is called is called anonymous class

It is used when you have to override method of class or interface, without having to actually subclass a class to make your code more concise.

```
MyClass m = new MyClass(){
        //logic
};
```

* Generic Programming

~~the~~ used for implementing iterable collection containing multiple items of any given type

Iterable Supports iteration (interface)

↓

Collection Supports addition (interface)

(interface) list support indexing

(interface) Set support identificat^n

class

ArrayList	linkedlist		hashset	treeset
O(1)	O(1)	add^n	O(n)	O(logn)
O(1)	O(n)	indexing	×	×
O(n)	O(n)	Searching	O(n)	O(logn)
1	3	ret/entry	1	3

Date:

list :- They are Sequential collection values are stored in the same order as insertion. duplicate values are allowed eg. age

set :- values are not stored in same order as insertion. duplicate values are not allowed eg rollno.

map :- Map is type of set whoes each item is key-value pair. here also duplicate values are not allowed since it is also a set.

* List

1) Arraylist

It grows ~~on the~~ automatically doesn't have to mention size

2) linked list

Same as arraylist but expensive in the terms of memory and performance.

* ~~Map~~ Set

1) Hashset

hashset uses hashcode() and equals() methods of object class to perform operations like put() and get(). that's why it doesn't support duplicate elements cuz ~~most~~ elements are stored ~~at~~ by checking hashcode if it is same value will be stored at same position.

2) Treeset

Treeset is implemented as Binary search tree :- thats why Treeset always give ~~sorted~~

~~sh~~ hashset and maps doesn't ~~allow~~ show duplicate it get overwritted by new value.

values in sorted order

Comparable & comparator

Comparable	Comparator
1) We can sort collection on the basis of single element such as id, name & price	We can sort collection on the basis of multiple elements such as id, name and price etc.
2) actual class is modified	actual class is not modified
3) provides compareTo()	provides compare() method
4) present in java.lang	present in java.util
5) May violate SRP if its not applicable to all	doesn't violate SRP as we are creating collection of objects & applying comparator

Comparator can be used if want to compare according to our logic eg Students according percentage or according to roll no. by ~~ou~~ implementing Comparator interface and overriding its method compare()

* Map

Map has key: value pair

duplicates are not allowed cuz, ~~key~~ duplicates keys are not allowed in a map. even if we give duplicate it will be ~~over~~ overwritten by new value.

Map uses entryset() method to iterate via map.

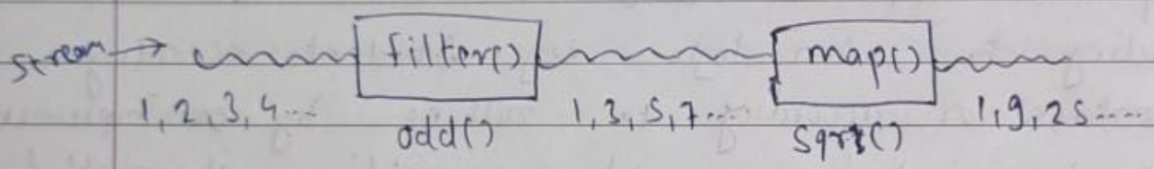
entryset() provides reference to a mapentry() which helps you to iteratere (visit) each value.

function taking function as argument is called Callback.

1) Hashmap
allows null value & null keys

2) Treemap
maintains order of the object but hashmap will not. Treemap cannot have null key but can have multiple null value

* Functional Programming
- data is processed by passing it through chain of calls (conveyer belt)
- To maintain generality it accepts another function of particular type as an arguments called as callback
- It also keeps original data safe ~~and modify~~



In filter() method we are passing Odd() as method, we can also pass even(), prime() ... so that generality will be maintained

In map()