5th August 2023

- Comparable interface
- Comparator interface

used to sort custom class's object

In java's classes like Integer, String, Comparable interface is already implemented

Q. How to sort custom class's object?

→ By using Comparable and Comparator interface, we can sort custom class's object.

```
public class Student
{
  private int rollno;
  private ~~[illegible]~~ String name;
  getter - —
  setter —
}
```

Exception in this code

```
public class Test
{
 public static void main(String[] args)
 {
  Set<Student> s = new TreeSet<>();
  Student stu1 = new Student();
  stu1. setRollno (2);
  stu1. set Name ("aaa");
      Student stu2 = new Student();
      stu2. setRollno (1);
      stu2. setName ("ccc");
     Student stu3 = new Student();
      stu3. setRollno(3);
      stu3. setName ("bbb");
   s. add (stu1);
   s.add (stu2);
   s. add (stu3);
```

Doubt - After which line is there exception?

```
for(Student stu : s)
{
      System.out.println( stu.getRollno());
      System.out.println( stu.getName());
}
}
}
```

This code will generate exception
class cast exception

```
public class Student implements Comparable<Student>
{
    private int rollno;
    private String name;
    public int getRollno() { return rollno; }
    public String getName() { return name; }
    public void setRollno(int rollno){ this.rollno = rollno; }
    public void setName (String name) { this.name = name;}
    @ Override                                  add() is calling
    public int compareTo(Student o)             compareTo()
    {
        return this.rollno - o.rollno;
     // for non-primitive type sorting ~~String name~~ - return this.name.compareTo(o.name);
    }
```

returns
0
-1
+1

Comparable - for default sorting
Comparator - for customized sorting

add()
is calling
compareTo()

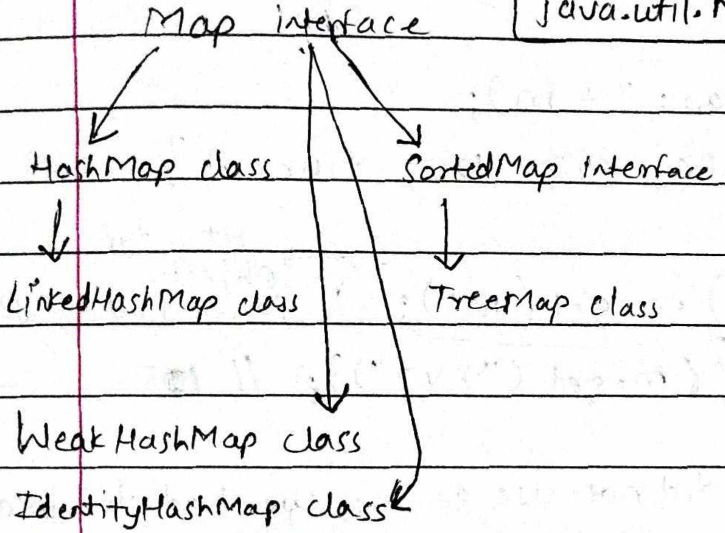
compare()
in
comparator

```
public class Test {
    p.s.v.m(String[] args)
    {
    Scanner sc = new Scanner(System.in);
    Set<Student> s = null;
    System.out.println("1. Roll no sort" +
                       "2. Name sort");
    int ch = sc.nextInt();
    if(ch == 1)
    {
    Comparator<Student> crno = (o1,o2)-> o1.getRollno() - o2.
                                                   getRollno();
    s = new TreeSet<Student>(crno);
    }
    else if (ch == 2)
    { Comparator<Student> crnm
       (o1, o2) ->
        o1.getName()
         .compareTo(o2.getName());
       s = new TreeSet<Student>
                         (crnm);
    }
```

Comparable	Comparator
① used for default sorting	① used for customized sorting
② Has one method - compareTo()	② 2 methods - compare(), equals()
③ Needs to be implemented in same class whose Object is to be sorted.	③ ~~Needs is not to be implemented~~ Not necessary to implement in same class.
④ from java.lang package	④ from java.util package

```
public class Test
{
    p.s.v.m ( String[] args)
    {

        Scanner sc = new Scanner (System.in);
        Set<Student> s = null;
        System.out.println("1. Rollno. sort" + "2. Name Sort");
        int ch = sc.nextInt();
        if(ch==1)                                  call to compare() method
        {
            Comparator<Student> crno = (o1,o2) ->
                                    o1.getRollno() - o2.getRollno()
                                                                    ;
            s = new TreeSet <Student> (crno);
        }

        else if ( ch == 2)
        {
            Comparator<Student> crnm =
                    (o1, o2) -> ~~o1.getRollno()~~
                            o1.getName().compareTo(o2.getName());

            s = new TreeSet<Student>(crnm);
        }

        Student s1 = new Student();
        s1.setRollno (2);
        s1.setName("ccc");
    Student s2 = new Student();
     s2.setRollno( 3);
     s2.setName( "bbb");
     s.add(s1);
     s.add (s2);
  }
}
```

java.util.Map



Map is ~~nott~~ **not** a part of Collection framework

Using generic, homogenous data can be stored

~~When to use map?~~ About Map:-

→ (1) store the data in key & value pairs

(2) Heterogenous data can be stored in map

(3) duplicate key is not allowed.

(4) duplicate value can be stored.

(5) ~~put()~~ - used to add data to map
put(key, value)

(6) get(key) - to get value from map

* (7) If we store duplicate key in map, value will be overridden
latest value will be stored against that key.

* (8) null as a key can be stored in Map (only once)

(9) null as value can be stored multiple times

(10) key-value pair = Entry

Uses of map

① while configuring database connection properties

Without generics

```
Map m = new HashMap();
m.put( 101, "PQR");
m.put(102, "ABC");
m.put("XYZ", 105);
m.put (true, 'A');
System.out.println("Map: " + m);
// {101 = PQR, 102 = ABC, XYZ = 105, true = A}
```

HasMap() - insertion order is not preserved

Insertion order is decided ~~as per hash code~~

```
String val
[illegible] [illegible] = (String) m.get(101);
System.out.println (m.get ("XYZ")); // 105
```

return type is Object. So, we had to typecast.

As we did not use generic, we had to typecast.

Without generics, Map m = new HashMap<>();
m.put(Object key, Object value);

With generics

```
Map<Integer, String> m = new HashMap();
m.put(1, "ABC");
m.put(2, "PQR");
Set<Integer> keys = m.keySet(); // [ , , ]
System.out.println( keys);
Iterator<Integer> itr = keys.iterator();
 while (itr.hasNext())
{
      int key = itr.next();
      String value = map.get(key);
      System.out.println( key + " " + value);
}
```

Keys are unique. keySet() has return type set.

Without generics -
HashMap - heterogenous data can be stored

or
```
for(int key: keys)
{
        String val = m.get(key);
        System.out.println(v);
}
```

HashMap - insertion order is not preserved
LinkedHashMap → insertion order is preserved
(remaining all properties are same as HashMap)

```
Map<String, Integer> map = new LinkedHashMap();
 map.put("ABC", 101);
 map.put ("PQR", 103);
 map.put("XYZ", 104);
 System.out.println(map);
 // {ABC=101, PQR=103, XYZ=104}
```
↑ insertion order is preserved

TreeMap - insertion order is not preserved
- default sorting is provided according to key.
  number - ascending order
  string - dictionary order or alphabetical order
- If null as key is stored in TreeMap → NullPointerException
- If heterogenous key → ClassCast Exception

```
~~Map<String, Integer> map = new TreeMap();~~
→ Map map = new TreeMap();
```
Without generics
```
map.put("ABC", 101);
map.put("PQR", 103);
map.put("GHI", 102);
map.put(true, 106);
```
← ClassCastException (for TreeMap)

Descending order
```
Comparator<String> c = (s1,s2) → s2.compareTo(s1);
Map<String,Integer> map = new TreeMap(c);
 map.put("ABC", 102);
```

①
```
Map<Integer, String> m1 = new HashMap<>();
m1.put(4567, "Audi");
m1.put( 798, "BMW");
m1.put(null, null);
Set<Integer> s1 = m1.keySet();
```
— Printing the values using Iterator will not be Exception if key is null

— Printing the values using enhanced for loop – will be Exception if the key is null.

I think, this is because Iterator<Integer>
↑ can have null

But, for(int key : set)
↑ cannot be null.

6th August 2023

Database

DBMS – to work with database

- ~~To~~ To store large amount of data in organized manner
- Types of databases – 1. Relational databases (SQL databases)
  2. Non-relational databases (No-SQL databases)

1. Relational databases – data in the form of table (rows & columns)
   - every row = one record
   - every column = one field
   - we can create relations between two or more tables.

MongodB – key-value ~~स्वरूप~~ data store होता.

relations betn 2 or more tables – 1 to 1, many to 1, 1 to many, many to many

Relational databases – Oracle DB, MySQL, PostgresQL, MS SQL

- To work with relational database, we've to use SQL query language. SQL (Structure Query Language)

Date: 06/08/2023

CRUD operations can be performed using SQL

C - Create or save data

R - Retrieve or fetch or get data from database

U - Update the data

D - Delete the data from database

SQL query language - not case sensitive ✓

Download MySQL installer 8.0 → Start download

- MySQL ~~workbench~~ command line client - Enter password
- Workbench - Public URL - localhost:3306,

- Schemas on left side, refresh button

Command prompt font size change -

Right click → Properties → font size can be changed at the top

Queries

1. Create database

```
Create database databaseName;
```

We can write the SQL queries in both mysql CLI & mysql workbench

MySQL workbench has GUIs & is userfriendly.

2. `show databases;`

To view all the databases

3. To drop database - `drop database databaseName;`
4. Select the database - `use databaseName;`

(After ~~the~~ executing this command, we can work in the database)

In mysql workbench, you've to select the query & then execute

MySQL datatypes

1. char – max. capacity 0 to 255
2. varchar – max. capacity 0 to 65535
3. int – max. capacity 255
4. for varchar, we've to specify the size e.g. varchar(25)

MySQL :- declaration – variableName dataType

Table related queries –

1. to create table –

```
create table tableName
(      column1 datatype,
       column2 datatype,
       .
);
```

2. to view tables – show tables;
3. to get information of table – ~~describe~~ desc tableName;
4. To drop table – drop table tableName;

table drop केल्यावर ते गेले आहे हे बघण्यासाठी show tables; वापरू शकतो.

CRUD related queries –

1. Create/save data in table –

- insert into student (rollno, name, age, marks) values (1, 'Raj', 25, 95.7);

- insert into student values (2, 'Swarali', 22, 98.76);

2. To retrieve all data from database

   select * from tableName;

③ To get particular data -

```
select * from student where name = 'Nayan';
```

④ delete particular data - delete from tableName where colName = value;

```
delete from student where rollno = 3 ;
```

⑤ update

```
update student set age = 25, marks = 90.50
        where rollno = 2 ;
```

```
update tableName set col1name = value, colName = val, - ---
where colName = val;
```