DAY -6

Day 6:Java Collection, Generics

- Collections Framework introduction
- List, Set, Map
- Iterator, ListIterator and Enumeration
- Collections and Array classes
- Sorting and searching, Comparator vs Comparable
- Generics, wildcards, using extends and super, bounded type
- Hands on & Lab

Object

- Object is an special class in java defined in java.lang
- Every class automatically inherit this class whether we say it or not…

We Writer like…

```
class Employee{
        int id;
        double salary;
        ....
        ....
        }
```

Java compiler convert it as…

```
class Employee extends Object{
        int id;
        double salary;
        ....
        ....
}
```

Why Java has provided this class?

rgupta.mtech@gmail.com

Method defined in Object class…

String toString()
boolean equals()
int hashCode()
clone()
void finalize()
getClass()

Method that can’t be overridden
final void notify
final void notifyAll
final void wait()

rgupta.mtech@gmail.com

toString( )

- If we do not override toString() method of Object class it print Object Identification number by default
- We can override it to print some useful information....

```
class Employee{
    private int id;
    private double salary;

    public Employee(int id, double salary) {
        this.id = id;
        this.salary = salary;
    }

}

        ......
        ......
        Employee e=new Employee(22, 333333.5);
        System.out.println(e);
        .....
        ....
```

O/P

```
com.Employee@addbf1
```

Java simply print object identification number not so useful message for client

rgupta.mtech@gmail.com

toString()

```
class Employee{
    private int id;
    private double salary;

    public Employee(int id, double salary) {
        this.id = id;
        this.salary = salary;
    }

    @Override
    public String toString() {
        return "Employee [id=" + id + ", salary=" + salary + "]";
    }

}
public class DemoToString {
    public static void main(String[] args) {
        Employee e=new Employee(22, 333333.5);
        System.out.println(e);
    }
}
```

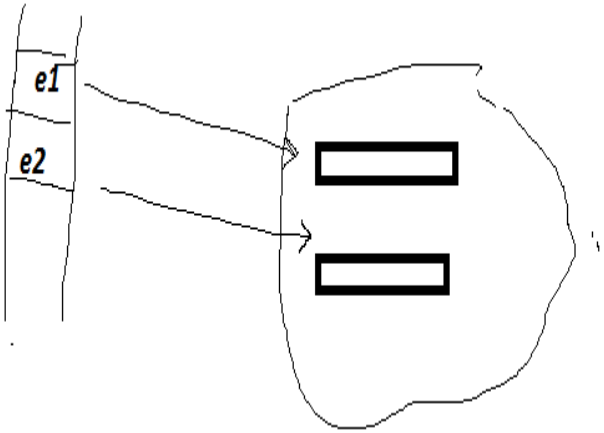
rgupta.mtech@gmail.com

equals

What O/P do you expect in this case……

```
Employee e1=new Employee(22, 333333.5);
        Employee e2=new Employee(22, 333333.5);

        if(e1==e2)
            System.out.println("two employees are equals....");
        else
            System.out.println("two employees are
```



O/P would be two employees are not equals.... ???

- Problem is that using == java compare object id of two object and that can never be equals, so we are getting meaningless result…

rgupta.mtech@gmail.com

Overriding equals()

Don’t forget DRY run.....

```
@Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Employee other = (Employee) obj;
        if (id != other.id)
            return false;
        if (Double.doubleToLongBits(salary) != Double
                .doubleToLongBits(other.salary))
            return false;
        return true;
    }
```

rgupta.mtech@gmail.com

hashCode()

Whenever you override equals()for an type don't forget to override hashCode() method…
hashCode() make DS efficient What hashCode does

HashCode divide data into buckets
Equals search data from that bucket…

```
@Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + id;
        long temp;
        temp = Double.doubleToLongBits(salary);
        result = prime * result + (int) (temp ^ (temp >>> 32));
        return result;
}
```

rgupta.mtech@gmail.com

clone()

- Lets consider an object that creation is very complicated, what we can do we can make an clone of that object and use that
- Costly , avoid using cloning if possible, internally depends on serialization
- Must make class supporting cloning by implementing an marker interface ie Cloneable

```
class Employee implements Cloneable{
    private int id;
    private double salary;

    public Employee(int id, double salary) {
        this.id = id;
        this.salary = salary;
    }

    @Override
    protected Object clone() throws CloneNotSupportedException {
        // TODO Auto-generated method stub
        return super.clone();
        //can write more code
    }

}
```

rgupta.mtech@gmail.com

finalize()

As you are aware ..Java don't support destructor Programmer is free from memory management
Memory mgt is done by an component of JVM ie called Garbage collector GC GC runs as low priority thread..
We can override finalize() to request java
"Please run this code before recycling this object" Cleanup code can be written in finalize() method Not reliable, better not to use…
Demo programm

WAP to count total number of employee object in the memory at any moment of time if an object is nullified then reduce count….

rgupta.mtech@gmail.com

Java collection

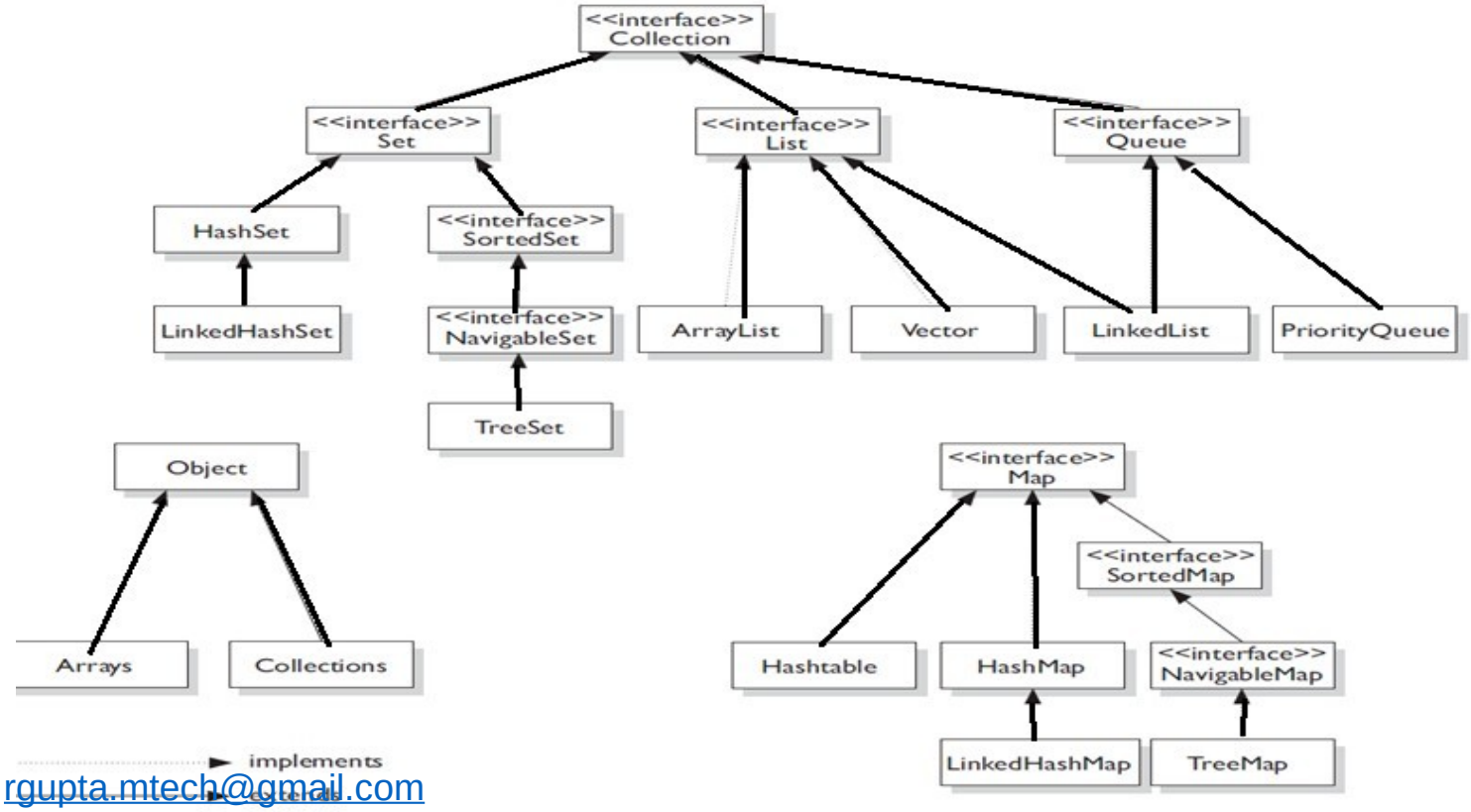
- Java collections can be considered a kind of readymade data structure, we should only need to know how to use them and how they work….

- collection
  - Name of topic
- Collection
  - Base interface
- Collections
  - Static utility class provide various useful algorithm

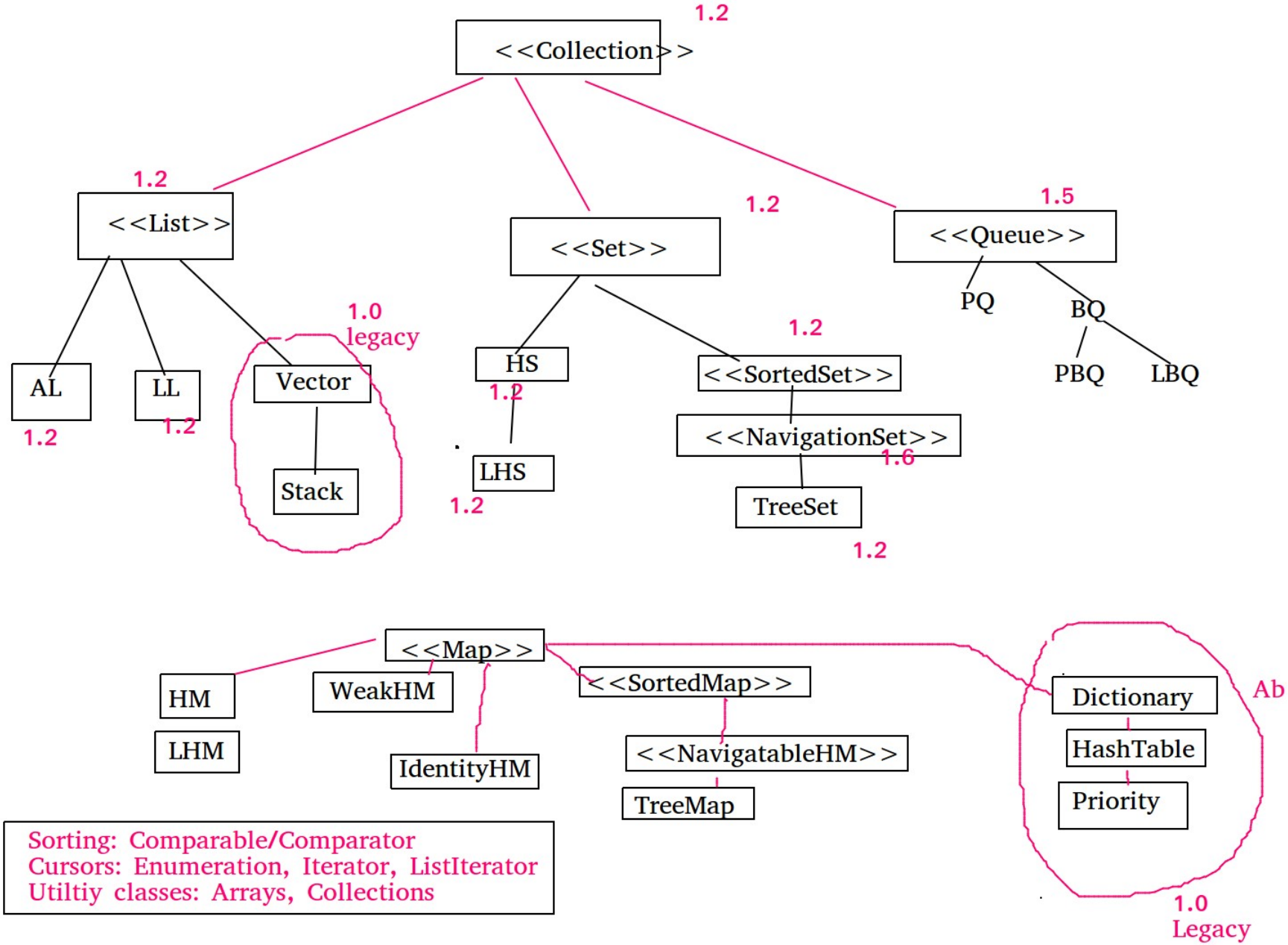
rgupta.mtech@gmail.com

collection

- collection
  - Why it is called an framework? Readymade Data structure in Java…



rgupta.mtech@gmail.com


<<Collection>>
1.2
<<List>>
1.2
<<Set>>
1.2
<<Queue>>
1.5
AL
1.2
LL
1.2
Vector
1.0
legacy
Stack
HS
1.2
LHS
1.2
<<SortedSet>>
1.2
<<NavigationSet>>
1.6
TreeSet
1.2
PQ
BQ
PBQ
LBQ
<<Map>>
HM
LHM
WeakHM
IdentityHM
<<SortedMap>>
<<NavigatableHM>>
TreeMap
Dictionary
HashTable
Priority
Ab
1.0
Legacy
Sorting: Comparable/Comparator
Cursors: Enumeration, Iterator, ListIterator
Utiltiy classes: Arrays, Collections

Four type of collections

Collections come in four basic flavors:

- **Lists** *Lists* of things (classes that implement List).
- **Sets** *Unique* things (classes that implement Set).
- **Maps** Things with a *unique* ID (classes that implement Map).
- **Queues** Things arranged by the order in which they are to be processed.

rgupta.mtech@gmail.com

Iterator in Java

```
List<String>list=new LinkedList<String>();
list.add("a");
list.add("b");

ListIterator<String> it = list.listIterator();
while(it.hasNext()){
    String val=it.next();
    if(val.equals("raj"))
        it.remove();
    else if(val.equals("a"))
        it.add("aa");
    else if(val.equals("b"))
        it.set("b1");
}
```

rgupta.mtech@gmail.com

ArrayList:aka growable array…

```
List<String>list=new ArrayList<String>();
...
...
list.size();
list.contains("raj");

test.remove("hi");

Collections.sort(list);
```

Note:

```
Collections.sort(list,Collections.reverseOrder());

Collections.addAll(list2,list1);
---------------------------
        Add all elements from list1 to end of list2

Collections.frequency(list2,"foo");
---------------------------------
        print frequency of "foo" in the list2 collection

boolean flag=Collections.disjoint(list1,list);
----------------------------------------
        return "true" if nothing is common in list1 and list2

Sorting with the Arrays Class
----------------------------
Arrays.sort(arrayToSort)

Arrays.sort(arrayToSort, Comparator)
```

rgupta.mtech@gmail.com

Iterator in Java

```
List<String>list=new LinkedList<String>();
list.add("a");
list.add("b");

ListIterator<String> it = list.listIterator();
while(it.hasNext()){
    String val=it.next();
    if(val.equals("raj"))
        it.remove();
    else if(val.equals("a"))
        it.add("aa");
    else if(val.equals("b"))
        it.set("b1");
}
```

rgupta.mtech@gmail.com

ArrayList of user defined object

```
class Employee{
        int id;
        float salary;
        //getter setter
        //const
        //toString
        }

        List<Employee>list=new ArrayList<Employee>();

        list.add(new Employee(121,"rama");
        list.add(new Employee(121,"rama");
        list.add(new Employee(121,"rama");

        System.out.println(list);

        Collections.sort(list);
```

How java can decide how to sort?

rgupta.mtech@gmail.com

Comparable and Comparator interface

We need to teach Java how to sort user define object
Comparable and Comparator interface help us to tell java how to sort user define object....

```
Comparable              Comparator
==========================================
java.lang                               java.util

Natural sort                            seconday sorts

Only one sort sequence                  as many as you want
is possible

need to change the                      Dont need to change
design of the class                     desing of the class


need to override                        need to override



public int                              public int
                                        compare(Employee o1, Employee o2)
compareTo(Employee o)
```

rgupta.mtech@gmail.com

Implementing Comparable

```
class Employee implements Comparable<Employee>{
    private int id;
    private double salary;
    .......
    .......
    .......

        @Override
    public int compareTo(Employee o) {
        // TODO Auto-generated method stub
        Integer id1=this.getId();
        Integer id2=o.getId();
        return id1.compareTo(id2);
    }
```

rgupta.mtech@gmail.com

Comparator

- Don’t need to change Employee class

```
class SalarySorter implements Comparator<Employee>{

    @Override
    public int compare(Employee o1, Employee o2) {
        // TODO Auto-generated method stub
        Double sal1=o1.getSalary();
        Double sal2=o2.getSalary();

        return sal1.compareTo(sal2);
    }

}
```

rgupta.mtech@gmail.com

Useful stuff

```
Converting Arrays to Lists
----------------------------------------

String[] sa = {"one", "two", "three", "four"};
List sList = Arrays.asList(sa);

Converting Lists to Arrays
----------------------------------------
List<Integer> iL = new ArrayList<Integer>();

for(int x=0; x<3; x++)
iL.add(x);

Object[] oa = iL.toArray(); // create an Object array

Integer[] ia2 = new Integer[3];

ia2 = iL.toArray(ia2); // create an Integer array
```

Arrays.binarySearch(arrayFromWhichToSearch,"to search"))

---

return -ve no if no found

array must be sorted before hand otherwise o/p is not predictiale

rgupta.mtech@gmail.com

Useful examples…

user define funtion to print the arraylist/linkedlist

```
printMe(list1);
....
.....


public void printMe(list<String>list){

        for(String s:list)
        {
                Sysout(s);
        }
}
```

user define funtion to remove stuff from a arraylist /linkedlist

---

```
removeStuff(list,2,5);



...
...

public void removeStuff(List<String>l, int from, int to)
{
        l.subList(from,to).clear();
}
```

rgupta.mtech@gmail.com

Useful examples…

Merging two link lists

```
ListIterator ita=a.listItrator();
Iteratory itb=b.iterator();

while(itb.hasNext())
{
        if(ita.hasNext())
                ita.next();

        ita.add(itb.next());
}
```

Removing every second element from an linkedList

```
itb=b.iterator();

while(itb.hasNext())
{
        itb.next();

        if(itb.hasNext())
        {
                itb.next();

                itb.remove();
        }
}
```

rgupta.mtech@gmail.com

LinkedList : AKA Doubly Link list… can move back and forth…….

```
Imp methods
--------------

        boolean hasNext()
        Object next()
        boolean hasPrevious()
        Object previous()

More methods
-------------
        void addFirst(Object o);

        void addLast(Object o);

        object getFirst();

        object getLast();

        add(int pos,Object o);
```

rgupta.mtech@gmail.com

Useful examples…

```
user define funtion to print linkedlist in reverse order
-----------------------------------------------------------

reversePrint(list);
...
....

public void reversePrint(list<String>l ){

        ListIterator<String>it=l.iterator(l.size());

        while(it.hasPrevious())
                Sysout(it.previous());
}
```

rgupta.mtech@gmail.com

fundamental diff bw ArrayList and LinkedList

- ArrayLists manage arrays internally. [0][1] [2][3][4][5] ....
  - List<Integer> arrayList = new ArrayList<Integer>();
  - LinkedLists consists of elements where each element has a reference to the previous and next element
    - [0]->[1]->[2] ....
    - <- <-

rgupta.mtech@gmail.com

ArrayList vs LinkedList

- Java implements ArrayList as array internally
  - Hence good to provide starting size
    - i.e. List<String> s=new ArrayList<String>(20); is better then List<String> s=new ArrayList<String>();
- Removing element from starting of arraylist is very slow?
  - list.remove(0);
  - if u remove first element, java internally copy all the element (shift by one)
  - Adding element at middle in ArrayList is very inefficient…

Performance Array List vs LinkedList

```
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

public class App
{
 public static void main(String[] args)
        {
        List<Integer> arrayList = new ArrayList<Integer>();
        List<Integer> linkedList = new LinkedList<Integer>();

         doTimings("ArrayList", arrayList);
         doTimings("LinkedList" , linkedList);
   }
```

```
private static void doTimings(String type, List<Integer> list)
  {
    for(int i=0; i<1E5; i++)
          list.add(i);
    long start = System.currentTimeMillis();

    /*
    // Add items at end of list
    for(int i=0; i<1E5; i++)
         {
      list.add(i);
    }
    */

    // Add items elsewhere in list
    for(int i=0; i<1E5; i++)
         {
      list.add(0, i);
    }
    long end = System.currentTimeMillis();

    System.out.println("Time taken: " + (end - start) + " ms for " + type);
  }
}
```

```
Time taken: 7546 ms for ArrayList
Time taken: 76 ms for LinkedList
```

rgupta.mtech@gmail.com

HashMap

Key ---->Value declaring an hashmap

- HashMap<Integer, String> map = new HashMap<Integer, String>();

Populating values

```
map.put(5, "Five");
 map.put(8, "Eight");
map.put(6, "Six");
map.put(4, "Four");
map.put(2, "Two");

      String text = map.get(6);

  System.out.println(text);
```

rgupta.mtech@gmail.com

Looping through HashMap

```
for(Integer key: map.keySet())
    {
        String value = map.get(key);

        System.out.println(key + ": " + value);
    }
```

most imp thing to remember

---

order of getting key value is not maintained

ie hashMap dont keep key and value in any particular order

rgupta.mtech@gmail.com

Other map varients

LinkedHashMap
Aka. Doubly link list
key and value are in same order in which you have inserted.....

TreeMap
sort keys in natural order(what is natural order?)
for int
1,2,3.....
for string
"a","b".....

For user define key
Define sorting order using Comparable /Comparator

rgupta.mtech@gmail.com

set

Don’t allow duplicate element

three types:

---

hashset
linkedhashset
treeset

HashSet does not retain order.

```
Set<String> set1 = new HashSet<String>();
```

LinkedHashSet remembers the order you added items in

```
Set<String> set1 = new LinkedHashSet<String>();
```

TreeSet sorts in natural order

```
Set<String> set1 = new TreeSet<String>();
```

Printing freq of unique words
from a file in increseing order of freq

---

words	freq
Apple	7
Ball	5
...	

rgupta.mtech@gmail.com

User define key in HashMap

If you are using user define key in HashMap do not forget to override hashcode for that class
Why?
We may not find that content again !

HashMap vs Hashtable

- Hashtable is threadsafe, slow as compared to HashMap
- Better to use HashMap

- Some more intresting difference
  - Hashtable give runtime exception if key is “null” while HashMap don’t

rgupta.mtech@gmail.com

HashMap vs Hashtable

- Hashtable is threadsafe, slow as compared to HashMap
- Better to use HashMap
- Some more intresting difference
- Hashtable give runtime exception if key is "null" while HashMap don't

rgupta.mtech@gmail.com

PriorityQueue

```
public class DemoPQ {
    public static void main(String[] args) {
        PriorityQueue<String> queue=new PriorityQueue<String>();
        queue.add("Amit");
        queue.add("Vijay");
        queue.add("Karan");
        queue.add("Jai");
        queue.add("Rahul");  //same as offer
        //retrieved not remove, throw exp
        System.out.println("head:"+queue.element());
        //retrieved not remove , return null
        System.out.println("head:"+queue.peek());

        System.out.println("iterating the queue elements:");
        Iterator<String> itr=queue.iterator();
        while(itr.hasNext()){
        System.out.println(itr.next());
        }
        //remove from head, throws ex if empty
        System.out.println(queue.remove());
        //remove from head, return null if empty
        System.out.println(queue.poll());

    }
}
```

rgupta.mtech@gmail.com

Generics

Before Java 1.5

- List list=new ArrayList();
- Can add anything in that list
- Problem while retrieving

Now Java 1.5 onward

- List<String> list=new ArrayList<String>();
  list.add("foo");//ok
  list.add(22);// compile time error

- Generics provide type safety
- Generics is compile time phenomena…

rgupta.mtech@gmail.com

Issues with Generics

- Try not to mix non Generics code and Generics code…we can have strange behaviour.

```
package com;
import java.util.*;

public class DemoGen1 {
    public static void main(String[] args) {

        List<String> list=new ArrayList<String>();
        list.add("foo");
        list.add("bar");

        strangMethod(list);

        for(String temp:list)
            System.out.println(temp);
    }

    private static void strangMethod(List list) {
        list.add(new Integer(22));// OMG......
    }

}
```

rgupta.mtech@gmail.com

Polymorphic behaviour

```
class Animal {
}

class Cat extends Animal{
}

class Dog extends Animal{
     }
```

```
Animal []aa=new Cat[4];// allowed

List<Animal>list=new ArrayList<Cat>()
```

rgupta.mtech@gmail.com

<? extends XXXXXX>

```
package com;
import java.util.*;

public class DemoGen1 {
    public static void main(String[] args) {

        List<Integer> list=new ArrayList<Integer>();
        list.add(22);
        list.add(33);

        strangMethod(list);

        for(Integer temp:list)
            System.out.println(temp);
    }

    private static void strangMethod(List<? extends Number> list) {
        list.add(new Integer(22));//Compile time error..... Good
    }

}
```

in strangMethod() we can pass any derivative of Number class but we are not allowed to modify the list

rgupta.mtech@gmail.com

<? Super XXXXX>

```
class Animal {
}

class Cat extends Animal{
}

class Dog extends Animal{
     }

class CostlyDog extends Dog{
}
     ........
     .........
          List<Dog>list=new ArrayList<Dog>();
          list.add(new Dog("white"));
          list.add(new Dog("red"));
          list.add(new Dog("black"));
          strangMethod(list);

     private static void strangMethod(List<? super Dog> list) {
          list.add(new CostlyDog());
     }

     .........
     ......
```

Anytype of Dog is allowed and can also modify list

rgupta.mtech@gmail.com

Generic class

```
class MyObject<T>{
    T myObject;

    public T getMyObject() {
        return myObject;
    }

    public void setMyObject(T myObject) {
        this.myObject = myObject;
    }

}
public class GenClass {

        public static void main(String[] args) {
            MyObject<String> o=new MyObject<String>();
            o.setMyObject(new Integer(22));
                                                  will not compile !!!
            //System.out.println(it.intValue());
        }
}
```

rgupta.mtech@gmail.com

Generaic method

```
class MaxOfThree{

    public static <T extends Comparable<T>> T maxi(T a,T b, T c){
        T max=a;
        if(b.compareTo(a)>0)
            max=b;
        if(c.compareTo(max)>0)
            max=c;
        return max;
    }

}
```

rgupta.mtech@gmail.com