Topics of discussion for upcoming sessions
a. Generics
b. Collection vs Collections
c. Disucssion on few utility methods
d. Comparable vs Comparator(I)
e. Stream API's
f. Date and Time API
g. enum
h. annotations
i. inner classes
j. fileoperations and io streams

I am still not understanding how you are using 2 methods in a single object.
eg: mapData.getClass().getName(); data.getKey().equals(10)

eg:
```
String name ="sachin";
String result = name.toUpperCase();
int count=result.length();
System.out.println(count);

          vs
String name ="sachin";
System.out.println(name.toUpperCase().length());
```

Q>Is e.printStack() and sysout(e) same?
Exception object
name of the exception
cause of the exception
line where the exception occured

e.printStackTrace() -> prints the name,cause and line where exception occured
System.out.println(e) similar to System.out.prinltn(e.toString()) -> name and cause of exception

Q> Why do we have to always declare System.out.println() inside a method?
Any statement should always be inside a method because of oop's style coding.

```
class NameOfTheClass{
            //properties(datatypes)

            //behaviours(methods)
                System.out.println("");
}
```

Q> Can we implement multithreading with abstract class and interface?
Ans. yes using interface only multithreading is implemented
eg: Runnable(I)

Q>

```
import java.util.*;
import java.io.*;
public class Properties {

    public static void main(String[] args) throws Exception{
```

```
            Properties p1=new Properties();
            FileInputStream fin=new FileInputStream("C:\\Users\\VLR\\Desktop\\
db.properties");

            p1.load(fin);//------->method load is undefined for type property
      }
}
```

Properties object is not created for java.util.Properties, rather it is created for userdefined class called "Properties".
solution : change the class name to PropertiesApp then run it will work.

Q>

```
package Arraylist;
import java.util.*;
public class IteratorArraylist
{
      public static void main(String[] args)
      {
            ArrayList a=new ArrayList();
            a.add(100);
            a.add(10);
            a.add(80);
            a.add(50);

            Iterator i=a.iterator();
            System.out.println("Acccesing data in Forward direction using
Iterator");
            while(i.hasNext())
            {
                  System.out.println(i.next());
            }

            ListIterator l=a.listIterator(a.size());
            //cursor is in last position
            System.out.println("Accessing data in Backward direction using
ListIterator");
            while(l.hasPrevious())
            {
                  System.out.println(l.previous());
            }
            System.out.println("Accessing data in Forward direction using
ListIterator");
            while(l.hasNext())
            {
                  System.out.println(l.next());
            }
      }
}
```

Sir, Here why it is necessary to pass size in ListIterator when we want to access data from last but in LinkedLlist it is not required?
  Even in LinkedList, we need to send size() return type otherwise the logic won't work.

```
package LinkedList;
import java.util.*;
public class IteratorLinkedList {
```

```
	public static void main(String[] args) {
		LinkedList l1=new LinkedList();
		l1.add(10);
		l1.add(20);
		l1.add(30);
		l1.add(40);

		System.out.println("---------------");
		Iterator i=l1.iterator();
		while(i.hasNext())
		{
			System.out.println(i.next());
		}
		System.out.println("---------------");

		ListIterator l=l1.listIterator();
		//here cursor is in first positions
		while(l.hasNext())
		{
			System.out.println(l.next());
		}

		//cursor already in last location
		System.out.println("---------------");
		while(l.hasPrevious())
		{
			System.out.println(l.previous());
		}
		System.out.println("---------------");
		Iterator d=l1.descendingIterator();
		while(d.hasNext())
		{
			System.out.println(d.next());
		}

	}

}
```

Sir can you please explain higher, ceiling, lower, floor method Sir

higher() -> This method returns the highest value

ceiling(100) -> This method returns the highest value including the supplied one

lower() -> This method returns the lowest value

floor(100) -> This method returns the lowest value including the supplied one

A ={100,101,102,103,104,105}

higher(102)-> 103

ceiling(102)-> 102

lower(102) -> 101

floor(102) -> 102

Q>

StringBuffer sb=new StringBuffer("ten");

```
String s=new String("ten");

System.out.println(s.equals(sb)); //false

//if(this == Object) //String == StringBuffer (false)
      retur true;
   else if ( this instanceof String)   // String instanceof StringBuffer(false)
      return true;
   return false;
```

Q>
Sir, multithreading pls,

```
public class Launch {
      public static void main(String[] args) {
            ThreadA obj = new ThreadA();
            Thread t = new Thread(obj);
            t.start();
// 2threads(user defined and main thread)
            synchronized (obj.total) {
                  obj.wait();
                  System.out.println(obj.total);
            }
      }
}
class ThreadA implements Runnable{
      public Integer total=0;
      @Override
      public void run() {
            synchronized (total) {
                  for(int i=0; i<5; i++) {
                        total = total + i;
                  }
                  total.notify();
            }
      }
}
```

1. why total should be made 0 or else it gives NullPointerException
2.Producer synchronize block why total only doesn't work and also with notify it doesn't work
3. consumer gives exception for obj.total and wait also together? Please correct the above code and explain...