OOPJ Notes Day-8 Date: 05/05/2023

ArrayStoreException

```
public class ArrayStoreDemo {

    public static void main(String[] args) {
        Object o1=new String("Malkeet");
        Object o2=new String("Saneep");
        Object o3=new String("Mahesh");
        Object o[]= new String[3];
        o[0]=new String("Malkeet");
        o[1]= new String("Singh");
        o[2]=new StringBuffer("Sandeep");

        //System.out.println(o1.toString()+" "+o2.toString()+" "+o3.toString());

for(int i=0;i<3;i++)
{
    System.out.println(o[i].toString());
}
    }

}
```

ClassCastException

```
class A
{
    int a;
}
class B extends A
{
    int b;
}

public class DemoOfClassCastEx {

    public static void main(String[] args) {

        A a1=new A();

        B b1=new B();

        B b2=(B)a1;

        b2.b=20;
```

```
        System.out.println("b is:"+b2.a);

    }

}
```

Difference between operator== and equals() method

- == it is operator which is used to compare value of two primitive data type.
- equals is method of Object class which is used to compare values of two objects of same class.
- user has to override this method to apply this inside user defined class.

```
public class DemoOfEqualsMethod {

    public static void main1(String[] args) {

        String s=new String("Malkeet");
        String s1=new String("Malkeet");

        //==

        if(s==s1)
        {
            System.out.println("Equal");
        }
        else
        {
            System.out.println("Not Equal");
        }

    }
    public static void main(String[] args) {

int s=20;
int s1=20;

        //==

        if(s==s1)
        {
            System.out.println("Equal");
        }
        else
```

```
        {
            System.out.println("Not Equal");
        }

    }

}
class Employee
{
    int EmpId;
    String Name;


    Employee(int EmpId, String Name)
    {
        this.EmpId=EmpId;
        this.Name=Name;
    }

    @Override
    public boolean equals(Object o) {

        if(o!=null)
        {
            if(o instanceof Object)
            {
                Employee oe=(Employee)o; //Downcasting

                if(this.Name==oe.Name && this.EmpId==oe.EmpId)
                {
                    return true;
                }
            }
        }
        return false;

    }
}

public class DemoOfEqualsMethodUserDefinedClass {

    public static void main(String[] args) {

        Employee e1=new Employee(1001,"Suresh");
        Employee e2=new Employee(1001, "Suresh");
```

```
        if(e1.equals(e2))
        {
            System.out.println("EQUAL");
        }
        else
        {
            System.out.println("Not Equal");
        }

    }

}
```

External system resources

- Following are the operating system resources that we can use for the application development:

1. Memory
2. Processor
3. Input and Output devices
4. File
5. Socket
6. Network Connections
7. Database connections
8. Operating System API

- In the context of Java, all above resources are non Java resources. These are also called as unmanaged resources(except memory).
- In the context of Java, resource is any external system resource that we can use in the application.
- Since operating system resources are limited, we should use it carefully. ### Exception Concept
- Do a brief study on this ### Throwable class Hierarchy
- Do a brief study on this ### Error versus Exception
- Do a brief study on this ### Checked versus unchecked exception
- Do a brief study on this ### AutoCloseable & Closeable interface and resource type
- Do a brief study on this ### Exception handling using try, catch, throw, throws and finally
- Do as much as practice you can

```
public class ExceptionDemo {

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

```
        int a=10;
        int b=5;
        int c=0;
        try
        {
            c=a/b;
        }

        catch (RuntimeException ex) {

            System.out.println("Exception Catched:"+ex.getMessage());
        }

        finally {
            System.out.println(" I am Finally");
        }


        System.out.println("Result:    "+c);


        System.out.println("Hello Am Executing");

        System.out.println("Program Stopped Norammly......");


    }

}
public class DemoOfThrow {


    public static void show()
    {
        System.out.println("Am show method");
        throw new ArithmeticException("Arithmatic Exception thrown from Show");
    }

    public static void main(String[] args) {

        try
        {
            DemoOfThrow.show();
        }
        catch(ArithmeticException ex)
        {
```

```
            System.out.println(" Catch Block:   "+ex.getMessage());
        }
        finally {
         System.out.println("Exception was there");
        }


        System.out.println("Am end of Main");

    }

}
```

try with resource

- will discuss tomorrow ### Custom exception and its need.
- will discuss tomorrow ### Factory class and factory method
- will discuss tomorrow