Java - Recursion and an Intro to OO

- Exceptions
- Big F and Small f in float
- Intro to Recursion
- Intro to OO


```
class test {
    public  static void main(String[] args) {
        double a;
        float b = (float) 3.14;
        System.out.println(b);
        String number = "3.141592653589";
        a = (new Double(number)).doubleValue();
        System.out.println(a);
    }

}
```


What happens now?

```
class test {
    public  static void main(String[] args) {
        double a;
        String number = "3.1dae41592653589";
        a = (new Double(number)).doubleValue();
        System.out.println(a);
    }
}
```


What has gone wrong?

First look at the documentation for `Double` found via search on `www.java.sun.com`

Double

`public Double(String s) throws NumberFormatException`

Constructs a newly allocated Double object that represents the floating- point value of type double represented by the string. The string is converted to a double value as if by the valueOf method (I-§1.6.22).

Parameters: s - a string to be converted to a Double

Throws NumberFormatException (I-§1.41) If the string does not contain a correctly formated number.

Catching Exceptions

When a function/method want to signal some error has occurred it throws an exception. If you don't want your program to simply stop, then you must catch that exception by using a `try` block together with a catch statement. You can have as many statements in the try block.

An exception is simply an object, some them have methods you can query to find out additional information about the error.

Objects

In java every thing is a an object except `int, float, double,short,byte,char`.

Because every thing else is an object you can define methods that take objects as parameters:

```
  public void add_obejct(Object o) {
  ....
 }
```

Later we will see examples of this, when we look at lists and trees.

The classes Float,Double , Int , ... are special versions of `float, int` , ... which are classes. Also they are a good place to include various library functions.

Mostly people use the `float,double , ....` for efficency.

Catch the exception

```
class test {
    public  static void main(String[] args) {
        double a;
        String number = "3.1dae41592653589";
        try {
            a = (new Double(number)).doubleValue();
            System.out.println(a);
        } catch ( NumberFormatException e) {
            System.out.println("Not a number ");
        }

    }
}
```


Throwing your own Exceptions

```
public void myMethod(String name) throws InvalidNameException {
  if (name == null) then throw InvalidNameException();
  .....

}
```

Then if you use this method you must catch the exception.

```
try { myObject.myMethod(myName); }
 catch (InvalidNameException e) {
               // Handle the exception here
    }
```

It is always good to think about error conditions.

Recursion

- How do we add up the series with just 1?
- How do we add up the list $(1 + 2 + \cdots + n - 1) + n$

Why use Recursion?

- Some languages don't have loops.
- Although numbers are pretty simple data structures with more complicated algorithms recursion is really the only simple way of expressing the algorithm.

Programming maxim :- First get it right, then get it efficient.

Recursion

- Or breaking Bigger problems into Smaller problems -

How do add up the numbers $1 + 2 + 3 + 4 + \cdots + n$?

One way use a loop.

Another way use Recursion.

Java Solution

```
class test {
    public static int add_n(int n) {
        if (n==1)
            return(1) ;
        else
            return (n+add_n(n-1));
    }
    public  static void main(String[] args) {
        int sum_10 = add_n(10);
        System.out.println(sum_10);
    }
}
```

Towers of Hanoi

```
class hanoi {
    public static void move(int count,int start , int finish, int te
        if(count >0) {
            move(count-1,start,temp,finish);
            System.out.print("Move disk ");
            System.out.print(count);
            System.out.print(" from ");
            System.out.print(start);
            System.out.print(" to ");
            System.out.println(finish);
            move(count-1,temp,finish,start);
        }
    }
```



```
    public  static void main(String[] args) {
        move(3,1,3,2);
    }
}
```

Try it with 64 pegs.

Intro to OO

- OO Thinking- Think about the data you have in the program and what operations can be applied to it.
- Package things up in a class.
- Use inheritance to reuse definitions.

OO example

```
class Point {
    int x , y;
    public void plot() {
        System.out.print("x="); System.out.println(x);
        System.out.print("y="); System.out.println(y);

    }
    public void set(int new_x,int  new_y) {
        x = new_x; y = new_y;    }
    public void translate(int x_s,int y_s) {
        x += x_s; y += y_s;    }
}
```

```
class point_test {
    public static void main(String[] argv) {
        Point p = new Point();
        p.plot();
        p.set(10,23);
        p.plot();
        p.translate(3,2);
        p.plot();
    }
}
```


Constructors

Some times you want to initialise an object with a value when you create it.

Java (as well as other OO languages) provide the concept of a class constructor.


```
class Point {
    int x , y;
    public Point(int n_x, int n_y) {
        x=n_x; y=n_y;
    }
    public void plot() {
        System.out.print("x=");
        System.out.println(x);
        System.out.print("y=");
        System.out.println(y);

    }
}
```



```
class point_test {
    public static void main(String[] argv) {
        Point p = new Point(3,4);

        p.plot();

    }
}
```