

Object-Oriented Programming:

13/11/2015

class: Template for creating object (cookie cutter!)

↳ we will define them in terms of Data + Methods.

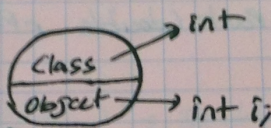
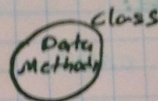
For ex: Die class

Face value → Data

Throw) → Method
Print

Concepts:

- 1) Encapsulation: we put things in class like hide or show
↳ can be private or public
- 2) Inheritance: we create things out of things.
- 3) Polymorphism ← dynamic binding



Constructors: (to create objects)

Student class;

// in each method we have constructors.

Student stu1 (10, "Ali");

" stu2 (

Destructors: For killing object (~)

C++ ✓

// we need space, allocate space for object, but

Java X

sometimes we need to allocate space, we need to kill

object to open new area.

Garbage Collection:

we need to do garbage collection else "Memory Leakage"

↳ We allocate storage we don't clear it back.

for (int i=1; i<11; i++)

{

...

}

} Allocate Memory!

Complex Numbers' Class:

18/11/2016

```
public class ComplexNo {
```

```
    private double realP, imagP; // instance variables, private means that: not accessible to outside
```

// we need constructor, and its name should be the same with class name

```
    public ComplexNo(double realPart, double imagPart) {  
        realP = realPart;  
        imagP = imagPart;  
    }
```

contru.

main method:

```
    ComplexNo cNo1 = new ComplexNo(5.0, 2.0); // cNo1 = 5.0 + i.20
```

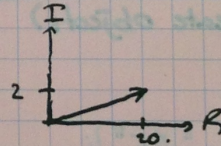
```
    public void setRealPart(double rPart) {
```

```
        realP = rPart;
```

```
    }
```

main method:

```
    cNo1.setRealPart(20.0);
```



```
    public void setImagPart(double iPart) // mutator.
```

```
    {  
        imagP = iPart;
```

```
    }
```

```
    public double getRealPart()
```

```
    {  
        return (realP);
```

```
    }
```

```
    public double getImagPart()
```

```
    {  
        return (imagP);
```

```
    }  
    public void addNoTo(ComplexNo other)
```

```
    {  
        realP = realP + other.realP;
```

```
        imagP = imagP + other.imagP;
```

main:

```
    cNo1.addOnTo(cNo2);
```

10+2i

After execution
15+15j

32

Main:

13/11/2018

CNo1.addOnTo(CNo2);

//write a method ~~for~~ for conjugate complex CNo3

```
public complexNo conjugate(){
```

```
    double rPart;
```

```
    double iPart;
```

```
    rPart = realP;
```

```
    iPart = imagP;
```

```
    ComplexNo newNo = new ComplexNo(rPart, iPart);
```

```
    return (newNo);
```

```
}
```

```
public String toString(){
```

```
    String result;
```

```
    result = Double.toString(realP) + "+" + Double.toString(imagP) + "i";
```

```
    return (result);
```

```
}
```

33