

Topics



- What are Mutable and Immutable Objects?
- Accessor and Mutator Methods ?
- How to make a class Mutable ?

Mutable and Immutable Objects



- Mutable Object → Object whose state (attribute values) can be changed after creation
- Immutable Object → Object whose state (attribute values) can not be changed after creation
- Immutable Objects read only objects.
- Immutable Objects can be freely shared among various clients

What are Accessor Methods?

- Accessor Method accesses/gets/reads the values of object's attributes especially when their Access Modifier is private
- Each Object Field / Instance Field / Attribute can have its own accessor Method

- General Preferred Form

`public <return-type> get<Instance-Field-Name>()
{
 return this.<Instance-Field-Name>;
}`

Generally Accessor Methods have public scope

Type of the instance-field for which accessor method is supplied

How to Name an Accessor Method



- In Java, the instance-fields are generally named in lower-case letters (For Example 'name' , 'age' , 'sex' etc.)
- To name an accessor method for any field, insert any of the words 'get' / 'read' before 'instance-field' name by making the first letter of the 'instance-field' name as capital
- For Example, if the instance-field is 'name' then its corresponding accessor method name can be getName() / readName()
- For Example, if the instance-field is 'age' then its corresponding accessor method name can be getAge() / readAge()
- For Example, if the instance-field is 'address' then its corresponding accessor method name can be getAddress() / readAddress()
- 'get' is used as a preferred word for accessor methods in the examples shown in the succeeding slides

Accessor Methods : Example 1



```
class Box
{
    private      double    length;
    private      double    width;
    private      double    height;

    // Accessor Method for length
    public              double    getLength()
    {
        return this.length;
    }
    // Accessor Method for width
    public              double    getWidth()
    {
        return this.width;
    }
    // Accessor Method for height
    public              double    getHeight()
    {
        return this.height;
    }
    .....
} // End of class Box
```

Accessor Methods : Example 2

innovate

achieve

lead

```
class Student
{
    private      String      name;
    private      String      idno;
    private      boolean     isMale;

    // Accessor Method for name
    public          String      getName()
    {
        return this.name;
    }
    // Accessor Method for idno
    public          String      getIdno()
    {
        return this.idno;
    }
    // Accessor Method for isMale
    public          boolean     getIsMale()
    {
        return this.isMale;
    }
    .....
} // End of class Student
```



What are Mutator Methods?

- Mutator Methods change the state of an object by changing their attribute values
- Mutator Methods should have return type as 'void'
- To name a mutator for an instance field, simply insert a 'set' [or any other suitable word such as 'write' or 'update'] before the instance field by making the first letter of the instance field name to upper case
- For example for an instance field named 'x', the mutator method should be

```
public void setX(type a)
{
    this.x = a;
}
```

- Arguments to a mutator method should be the type of instance field

Mutator Methods : Example 1

```
class Box
{
    private      double  length;
    private      double  width;
    private      double  height;

    // Mutator Method for length
    public              void      setLength(double length)
    {
        this.length = length;
    }
    // Mutator Method for width
    public              void      setWidth(double width)
    {
        this.width = width;
    }
    // Mutator Method for height
    public              void      setHeight(double height)
    {
        this.height = height;
    }
    .....
} // End of class Box
```


Mutator Methods : Example 2



```
class Student
{
    private      String      name;
    private      String      idno;
    private      boolean     isMale;

    // Mutator Method for name
    public        void        setName(String name)
    {
        this.name = name;
    }
    // Mutator Method for idno
    public        void        setIdno(String idno)
    {
        this.idno = idno;
    }
    // mutator Method for isMale
    public        void        setIsMale(boolean isMale)
    {
        this.isMale = isMale;
    }
    .....
} // End of class Student
```

How to Make a class Immutable?



- Make the instance fields private
- Supply only Accessor Methods in the class for getting/reading the values of instance fields
- Do not supply any mutator method inside the class

Immutable class Example



```
class Circle
{
```

```
    private    double    radius;
```

```
    // Constructor Method
```

```
    Circle(double radius)
```

```
    {
```

```
        this.radius = radius;
```

```
    }
```

```
    // Accessor Method
```

```
    public    double    getRadius()
```

```
    {
```

```
        return this.radius;
```

```
    } // End of Method
```

```
    // Method to compute area
```

```
    public    double    area()
```

```
    {
```

```
        return 3.1456 * radius * radius;
```

```
    } // End of Method
```

```
    // Method to Perimeter
```

```
    public    double    perimeter()
```

```
    {
```

```
        return 2 * 3.1456 * radius;
```

```
    } // End of Method
```

```
} // End of class Circle
```



Step 1: Instance Fields private



Step 2: Only Accessor Methods



Step 3: No Mutaor Method

Thank You