Constructor Problems!!!

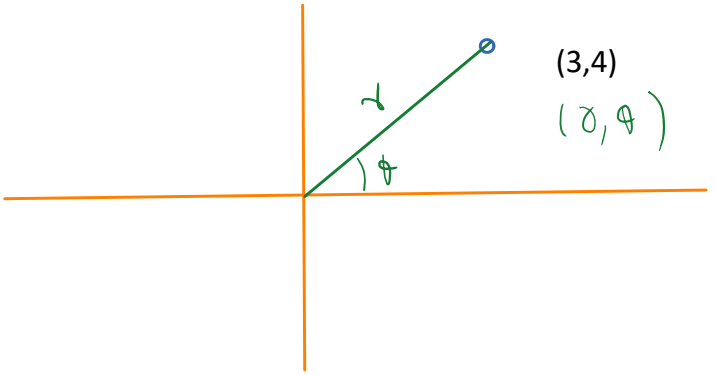
06 June 2018 15:03

1. Constructor is used with "**new**" and there is no old/existing keyword
   - There is a difference between needing an object and needing a new object
   - No object reuse
   - every time you call a constructor it's a new object

   Example:

   ```
   BankAccount x=new BankAccount(1); //new object
   BankAccount y=new BankAccount(2); //new object

   BankAccount z= new BankAccount(1);  //is this a new object??
   ```

2. Constructor has a completely meaningless name.
   - Class Name represents the Object
   - Constructor is the creator of the Object not a part of the object
   - A name valid for the object may not be valid for the creator of the object
   - Programmer **can't choose** the name to ensure it is meaningful.



3. Constructor is one of the few **non-polymorphic** methods that exist in OO design
   - polymorphism works if a base reference refers to a derived object
     - when constructor is called there is no object
   - constructors can't be virtual
   - constructors can't be overriddent
     - each class constructor varies in its name
   - constructors can't be replaced
     - The violate
       - OCP
       - LSP
   - constructors can't be specified in an interface
     - No dependency inversion
   - interface defines a contract which implementor must implement
     - Interface makes an objectresponsible to implement the standard
   - constructor is not a part of the interface
     - No standard can be forced on constructor
     - Not responsible.

4. Constructor has same name as that of the concrete class

- To create object you need to know constructor
- knowing constructor is knowning concerete class
- knowing concrete class is breaking Dependnecy Inversion Principle.

5. constructors are not optional.
    - constructors can't be removed.
    - constructors, however, can be hidden

Static Vs Non Static Members

06 June 2018 15:11

Feature	Static	Non-Static
owner	**class level**	**object level**
scope	compile time	runtime
memory	data segment	heap/stack
keyword	**static**	**n/a**
this pointer	**n/a**	**referes to the object**
how to access them	**ClassName as reference**	**object reference**

UserManagementSystem

06 June 2018 15:06

```
class UserUI{

    public  onRegistrationClick(){
        //A new user in the system
        User user=new
        InactiveUser( name,email,phone,password);
    }

    public onLoginClick(){
        //Is this also a new user in the system???
        User user=null;//new User(email, password);

        ResultSet set=query("select * from user…");
        if(set.Read()){
            switch(set["usertype"]){
                case "ADMIN": user=new Admin(…);
                break;
                case "CUSTOMER" : user=new Customer(…);
                …
            }
        }

    }


}
```

Change Ripples

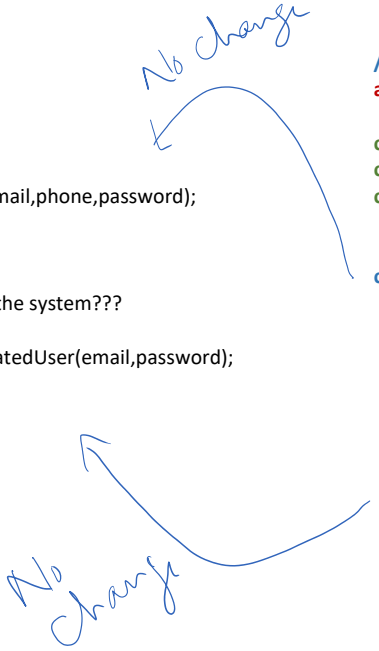
```
//Business Layer
abstract class User{}

class Employee: User{}
class Admin:User{}
class Customer:User{}

class InactiveUser:User{

    public InactiveUser(String name, String email, String phone,String password){

        //insert into users …
    }

    public User(String email, String password){
        //select from users …
    }

}
```

A data access code is written in presentation tier bypassing business layer

UserManagementSystem V2

06 June 2018 15:06

```
class UserUI{

    public  onRegistrationClick(){
        //A new user in the system
        User user= UserManager
            .RegisterUser( name,email,phone,password);
    }

    public onLoginClick(){
        //Is this also a new user in the system???
        User user= UserManager
                    .GetAuthetnicatedUser(email,password);

    }


}
```




```
//Business Layer
abstract class User{}

class Employee: User{}
class Admin:User{}
class Customer:User{}


class UserManager{
    public User RegisterUser(String name, String email, String phone,String password){

        //insert into users …

        return new InactiveUser(…);
    }

    public User GetAuthenticatedUser(String email, String password){

        User user=null;
        //select from users …
        ResultSet set=query("select * from user…");
        if(set.Read()){
            switch(set["usertype"]){
                case "ADMIN": user=new Admin(…);
                break;
                case "CUSTOMER" : user=new Customer(…);
                …
            }
        }


        return user
    }

}
```

Static Factory vs Constructor

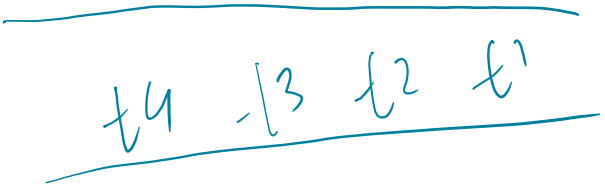
07 June 2018 10:58

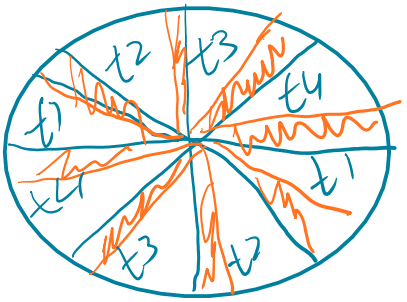
```
void main()
{

    RBI rbi = Government.GetRBI();   // new RBI();

    Bank icici = rbi.GetBank("ICICI");
    Branch iciciECBranch= icici.GetBranch("EC");
    BankAccount a1= iciciECBranch.OpenAccount(...);


}
```

How many RBI???

Thread

07 June 2018 11:23





C++ UIFactory Provider

07 June 2018 14:13

```
//steel.dll

class SteelForm : public UIForm {}
class SteelButton: public UIButton{}
class SteelTextBox : public UITextBox{}

class SteelFactory : public UIFactory{
	public UIForm * CreateForm() { return new SteelForm(); }
	public UIButton * CreateButton() { return new SteelButton(); }
	public UITextBox * CreateTextBox() { return new SteelTextBox(); }
}

UIFactory * GetFactory(){
	return new SteelFactory();
}


//Rubber.dll

class RubberForm : public UIForm {}
class RubberButton: public UIButton{}
class RubberTextBox : public UITextBox{}

class RubberFactory : public UIFactory{
	public UIForm * CreateForm() { return new RubberForm(); }
	public UIButton * CreateButton() { return new RubberButton(); }
	public UITextBox * CreateTextBox() { return new RubberTextBox(); }
}

UIFactory * GetFactory(){
	return new RubberFactory();
}




//core.dll

typedef UIFactory  (*factory) ();

class FactoryProvider
{
	public:
		UIFactory * GetFactory( char * themePath )
```

```
        {

            HANDLE dll = LoadLibrary(themePath) ; //linux --> dlLoad()
            if(dll==NULL)
                return new DefaultFactory();

            UIFactory factory=(UIFactory) GetProcAddress("GetFactory"); //linux --> dlSym()

            return factory();

        }

}
```

Encapsulation vs Inheritance Proxy

08 June 2018 08:54

```
//encapsulation version

class AuthenticatedSearchEngine : ISearchEngine
{
    ISearchEngine target;
    public AuthenticatedSearchEngine(ISearchEngine e){
        target=e;
    }

    public IResult Search(IQuery q){

        ...
        target.Search(q);
    }
}

//usage

var engine= new AuthenticatedSearchEngine(new BookSearchEngine());
```

```
//Inheritance version

class AuthenticatedSearchEngine : BookSearchEngine
{

    public IResult Search(IQuery q){

        ...
        base.Search(q);
    }
}

//usage

var engine= new AuthenticatedSearchEngine();
```

Electricity System

08 June 2018 11:12