STATIC DESIGN

BY

VIVEK DUTTA MISHRA

STATIC MEMBERS

- STATIC FIELDS
  - SINGLE COPY OF MEMORY ALLOCATED
  - SHARED WITH ALL OBJECTS OF THE CLASS
  - CAN BE ACCESSED BY BOTH STATIC AND NON STATIC METHODS
- STATIC METHODS
  - NO THIS POINTER
  - CAN ACCESS ONLY STATIC MEMBERS
  - CAN BE INVOKED USING CLASS REFERENCE RATHER THAN OBJECT REFERENCE
- ALL STATIC MEMBERS
  - CLASS LEVEL
  - THEY DON’T BELONG OBJECTS
  - DON’T REQUIRE OBJECT TO ACCESS THEM

IS THERE ANY STATIC CANDIDATE IN THIS CLASS?

```
  class BankAccount
  {
          String name;
          int accountNumber;
          double balance;
          String password;
static double rate;


          public void Deposit(double amount){
                  …
          }

          public void CreditInterest(){
                  balance+=(balance*rate)/1200;
          }
public static    void SetRate(double r){rate=r;}
  }
```

Every BankAccount will have its own name, blance passoword etc

But all account share same interest rate?

Why Do you Need Method to be Public?

So That we don't need an Object to call It.

STATIC CONSIDERATION

- STATIC IS CLASS LEVEL.
  - BUT CLASSES DOESN'T EXIST.
  - SO WHO OWNS THEM?
- STATIC MEANS OBJECT NOT REQUIRED
  - IS IT REALLY A OBJECT ORIENTED DESIGN?
- STATIC IS REFERENCED USING CLASS
  - SHOULDN'T IT BE CONSIDERED CLASS ORIENTED DESIGN?

IS THERE ANY STATIC CANDIDATE IN THIS CLASS?

```
 class BankAccount
 {
        String name;
        int accountNumber;
        double balance;
        String password;
static double rate;

        public void Deposit(double amount){
                …
        }

        public void CreditInterest(){
                balance+=(balance*rate)/1200;
        }
public static   void SetRate(double r){rate=r;}
 }
```

Does Interest Rate Belong to individual BankAccount Object?

Why is it Present in BankAccount class?

Who Owns Interest Rate?

IS THERE ANY STATIC CANDIDATE IN THIS CLASS?

```
class BankAccount
{
        String name;
        int accountNumber;
        double balance;
        String password;
        static double rate;

        public void Deposit(double amount){
                …
        }

        public void CreditInterest(){
                balance+=(balance*rate)/1200;
        }
        public static    void SetRate(double r){rate=r;}
}
```

```
class Bank{

        double rate;
        public void SetRate(double r){rate=r;}

}
```

Should I create a new class just to hold static members?

A closer look will confirm that your business Domain needs it

WHY DO I NEED BANK?

- WHO CREATES BANK ACCOUNT? DOES IT GET SELF CREATED?
  - EVERY OBJECT NEEDS A CREATOR OBJECT
- Bank CREATES BankAccount

- BANK MANAGES BankAccount
  - CREATOR AND MANAGER MAY NOT ALWAYS BE SAME

DEFINE THE BANK CLASS

```
class BankAccount
{
        String name;
        int accountNumber;
        double balance;
        String password;


        public void Deposit(double amount){
                …
        }

        public void CreditInterest(){
                balance+=(balance*rate)/1200;
        }

}
```

```
class Bank{
        double rate;
        public void SetRate(double r){rate=r;}
        List<BankAccount>  accounts;

        public int OpenAccount( … ){

                BankAccount a=new BankAccount(…);
                accounts.Add(a);
                return a.accountNumber;
        }

        public void Deposit( int accountNumber, int amount){
                GetAccount(accountNumber)
                        .Deposit(amount);

        }
}
```

USING BANK

```
void main(){

    Bank icici= …

    int account= icici.OpenAccount(…); //creates BankAccount but returns Account Number

    icici.Deposit( account, 20000);


}
```

WHY SHOULD YOU NOT USE STATIC?

- IN AN OBJECT ORIENTED DESIGN EVERYTHING (SHOULD) BELONG TO ONE OBJECT OR THE OTHER
- CLASS IS A DESCRIPTION OF OBJECT
    - IF IT IS NOT A PART OF THE OBJECT, IT SHOULDN'T BE PART OF THE CLASS
- STATIC IS A MECHANISM TO AVOID OBJECT
    - AVOID OBJECT ORIENTED DESIGN
- IN DOMAIN THERE IS NO ROOM FOR STATIC ELEMENT.
    - THEY CONTRADICT CLASS DOESN'T EXIST
        - ITS JUST A BLUEPRINT.

STATIC CONSIDERATION

- IS THERE ANY REAL PROBLEM WITH STATIC OTHER THAN CONTRADICTING PURIST OO VIEW?
- STATIC CANT BE VIRTUAL
- STATIC IS NON POLYMORPHIC
- STATIC DOESN'T SUPPORT LSP
- STATIC DOESN'T SUPPORT DIP
- SINCE STATIC CAN'T BE OVERRIDDEN THE NEED TO BE MODIFIED BREAKING OCP

FINAL THOUGHTS

- STATIC SHOULD BE AVOIDED
  - STATIC IS NOT OBJECT ORIENTED
  - STATIC VIOLATES KEY DESIGN PRINCIPLES – OCP, DIP, LSP
  - STATIC IS NON-POLYMORPHIC
- STATIC IS A LANGUAGE FEATURE
  - SEVERAL FEATURES DEPENDS ON STATIC
  - NOT COMPLETELY AVOIDABLE.

STATIC IS BAD NOT AS KEYWORD BUT AS THE NOTION IT BRINGS – “NO OBJECT”