Untitled

- Thread-Safe Singleton Pattern: Örnek: Hesap Bankalarında olan hesap Bakiyesi nesnesi için yaratılmak istenir ve yapılan değişiklikler(Para yatırma ,Para çekme işlemleri vb.) aynı nesne üzerinden yapılacaktır. Örneğin Kodları: hesapBakiye.java package Singleton_Pattern;

public class hesapBakiye {

```
private static hesapBakiye hesapBakiye_sample;
private double h_Bakiye;
```

```
protected hesapBakiye(){
 // Default Constructor
}
```

```
public static hesapBakiye CreateObject(){
 if (hesapBakiye_sample == null){
```

```
  hesapBakiye_sample = new hesapBakiye();
 }
 return hesapBakiye_sample;
}
```

```
public void setBakiye(double value){
 h_Bakiye = value;
}
public double getBakiye(){
 return h_Bakiye;
}
```

```
public boolean ParaCek(double cekilenPara){
 if(cekilenPara <= h_Bakiye){
  h_Bakiye = h_Bakiye-cekilenPara;
  return true;
 }else {
  return false;
 }
```

```
}
public boolean ParaYatir(double yatirilanPara){
 try{
  h_Bakiye = h_Bakiye + yatirilanPara;
  return true;
 }catch (ArithmeticException e){
  return false;
 }
```

```
}
```

} hesapBakiyeMain.java package Singleton_Pattern; import java.util.Scanner; public class hesapBakiyeMain {

```
private static Scanner scanner;
```

```
public static void main(String[] args) {
 scanner = new Scanner(System.in);
 hesapBakiye obj1 = hesapBakiye.CreateObject();
 hesapBakiye obj2 = hesapBakiye.CreateObject();
 hesapBakiye obj3 = hesapBakiye.CreateObject();
```

```
obj1.setBakiye(2000);
```

```
System.out.println("### Singleton Design Pattern's Example is Runing ###
\n");
```

```
System.out.printf("obj1-Bakiye : %.2f \n", obj1.getBakiye());
System.out.printf("obj2-Bakiye : %.2f \n", obj2.getBakiye());
System.out.printf("obj3-Bakiye : %.2f \n\n", obj3.getBakiye());
```

```
obj2.setBakiye(3000);
```

```
System.out.printf("obj1-Bakiye : %.2f \n", obj1.getBakiye());
System.out.printf("obj2-Bakiye : %.2f \n", obj2.getBakiye());
System.out.printf("obj3-Bakiye : %.2f \n", obj3.getBakiye());
```

```
System.out.println("\nYatırılacak Paranın Tutarını Giriniz Lütfen
 (Double Number) :");
double yatirilanPara = scanner.nextDouble();
```

```
if(obj2.ParaYatir(yatirilanPara)){
 System.out.printf("%.2f TL Hesabınıza Başarıyle
 Yatırıldı !!",yatirilanPara);
```

```
 System.out.printf("\nobj1-Bakiye : %.2f \n", obj1.getBakiye());
 System.out.printf("obj2-Bakiye : %.2f \n", obj2.getBakiye());
 System.out.printf("obj3-Bakiye : %.2f \n", obj3.getBakiye());
}else{
 System.out.println("Lütfen istenilen kriterlere uygun bir sayı
 giriniz !!");
}
```

```
}
```

} Result: # Singleton Design Pattern’s Example is Runing #

obj1-Bakiye : 2000,00 obj2-Bakiye : 2000,00 obj3-Bakiye : 2000,00

obj1-Bakiye : 3000,00 obj2-Bakiye : 3000,00 obj3-Bakiye : 3000,00

Yatırılacak Paranın Tutarını Giriniz Lütfen (Double Number) : 250 250,00 TL Hesabınıza Başarıyle Yatırıldı !! obj1-Bakiye : 3250,00 obj2-Bakiye : 3250,00 obj3-Bakiye : 3250,00

*UML Dİagram Class hesapBakiyeMain { +main():void } Class hesapBakiye { -hesapBakiye_sample:hesapBakiye +CreateObject():hesapBakiye +setBakiye(double):void +getBakiye():double +ParaCek(double):boolean +ParaYatir(double):boolean } hesapBakiye ⇓ hesapBakiyeMain :??? hesapBakiyeMain --- hesapBakiyeMain : New(Self Association)