Advanced Java

Miraç (Aralık-2022)

JAVA Generics

- Generics, farklı referans veri tiplerini alan, hangi referans tipini alacağına karar verebileceğimiz ve üzerinde benzer işlemler yapabileceğimiz bir sınıfdır.
- 2004'de Java 5 çıktığında beraberinde Generic gibi çok önemli özellikler geldi.
- Generic öncesi karşılaşılan problemlerin çözümü için Object ve List yapıları kullanıldı.
- Cast işlemi ve ClassCastException hatası

JAVA Generics-2

```java
// Interface ile kullanım
public interface Comparable<T> {
    public int compareTo(T object);
}

// Class ile kullanım
public class Printer<T> {
    public void print(T object) {
        System.out.println(object);
    }
}

// Method ile kullanım -1
public <T> T myGenericMethod(){
    T object = null;
    //***
    return object;
}

// Method ile kullanım -2
public <T> T myGenericMethodWithParameter(T object){
    return object;
}

// Method ile kullanım -3
public <T> void myGenericVoidMethod(T object){
    System.out.println(object);
}
```

Generics Öncesi :

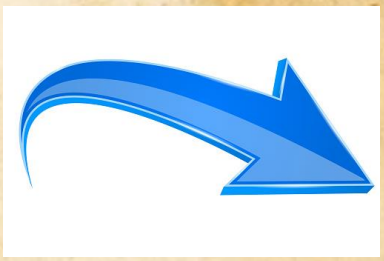
```java
// Object ile kullanım
List<Object> objectList = new ArrayList<Object>();
objectList.add("Merhaba Dünya");
objectList.add(123);
String string  = (String) objectList.get(0);
Integer integer = (Integer) objectList.get(1);

// Raw ile kullanım
List rawList = new ArrayList();
rawList.add("Merhaba Dünya");
rawList.add(123);
String string  = (String) rawList.get(0);
Integer integer = (Integer) rawList.get(1);
```

Generics - JAVA-7

- // Before Java 7
  - List<String> list = new ArrayList<String>();
- // After Java 7
  - List<String> list = new ArrayList<>();

JAVA Serialization

Nesnelerin içerisindeki değişkenlerin adları, tipleri değerleri byte'lara çevrilerek kaydedilmesi ve transfer edilmesi işlemi için kullanılan yapıdır.



JAVA Enum

- "**Sabit**" yapılar kurmaya yarar..
- Enum java için bir **keyword**'dür
- "**==**" ve "**equal()**" için aynı sonucu verir
- "**new**" ile yeni bir "**instance**" oluşturulamaz
- Varsayılan olarak "**static**" ve "**final**" tanımlıdır
- Enum constructor'ları daima "**private**" dır.

JAVA ENUM

```
enum Day{
    SUN, MON, TUE, WED, THU, FRI, SAT;
    public static void main(String args[])
    {
        System.out.println("This is enum");
    }
}
```

```
enum WeekDays
{
   sunday, monday, tuesday, wednesday, thursday, friday, saturday
}

class Test
{
  public static void main(String []er)
  {
    WeekDays wd;
    wd = WeekDays.sunday;
    System.out.println("Today is "+wd);
  }
}
```

Gender.java

```
1  package labsim.model.enums;
2
3  public enum Gender {
4      Male,
5      Female,
6
7  }
8
```

JAVA Threads

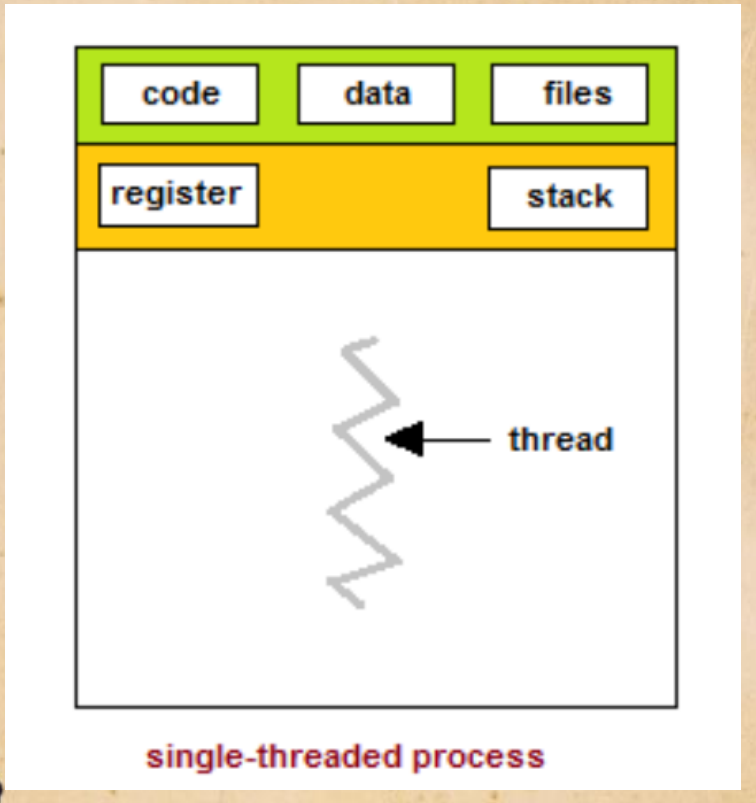
- Threadler aynı anda birden fazla işin yapılmasını sağlayan yapılardır.
- 1 processin parçasıdır, iş parçacığı da denilebilir
- Her thread kendi stack, program sayacını ve local değişkenlerini kullanırlar
- 1 processes ın altında birden fazla threads olabilir ve her thread bu processe ait memory (hafızasını) kullanır
- Her Java programında en az 1 threads kullanılır ; o da main thread.

Niye Thread Kullanırız ?

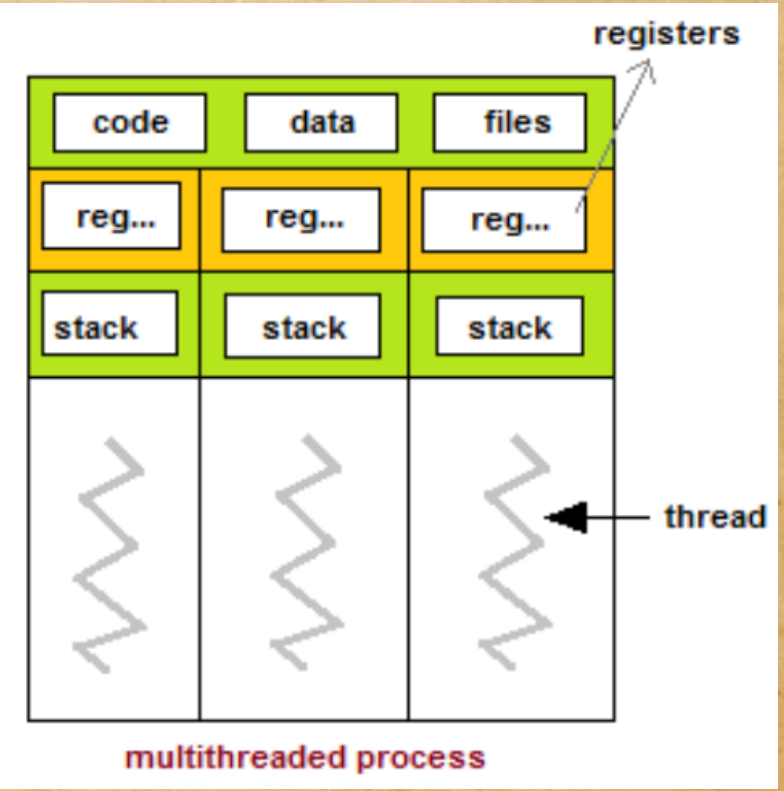
- UI ları daha responsive yapmamızı sağlıyor; dosya download ederken, web gezinme işleminin durmaması gibi..
- Multiprocessor sistemlerin avantajını kullanmak
- Background Processing ve Asynchronous

Niye Thread Kullanırız ?

- Single Thread



- Multi Thread



Multi-Thread bir yapıda , Data ortak ise ve aynı anda birden fazla thread aynı dataya ulaşmak ve değiştirmek isterse ne olur ???

Thread Nasıl Oluşturulur ? ?

- Thread Class
- Runnable Interface

First way to create and start a thread- Extends Thread Class

```
class MyThread extends Thread{

@Override
public void run() {
System.out.println("It is custom thread");
  }
}

public static void main(String[] args) {

      MyThread myThread =new MyThread();
      myThread.setName("ExampleThread");
      myThread.start();
```

Second way to create and start a thread- Implements Runnable Interface

```
class MyRunnable implements Runnable{

      @Override
      public void run() {
            System.out.println("Implemented Runnable interface");

      }
}

public static void main(String[] args) {

            Thread thread= new Thread(new MyRunnable());
            thread.start();
```