# Nesneye Yönelik Programlama BLM2012

Öğr. Grv. Furkan ÇAKMAK

## Ders Tanıtım Formu ve Konular

BLM2012 Nesneye Yönelik Programlama Hafta 12

| Hafta | Tarih | Konular |
|---|---|---|
| 1 | 01.03.2022 | Dersin ve Java Dilinin Genel Tanıtımı, Sınıflar, Nesneler, Üyeler, Final ve Static Kavramları |
| 2 | 08.03.2022 | UML Sınıf Şemaları, Kurucular ve Sonlandırıcılar, Denetim Akışı, Nesneleri Oluşturulması |
| 3 | 15.03.2022 | Kurucuların ve Metotların Çoklu Tanımlanması, İlkeller, String ve Math Sınıfları |
| 4 | 22.03.2022 | Sahiplik ve Kullanma İlişkileri, Tek Yönlü ve İki Yönlü Sahiplik Kavramları |
| 5 | 29.03.2022 | Kalıtım, Metotların Yeniden Tanımlanması ve Çoklu Metot Tanımlamadan Farkı |
| 6 | 05.04.2022 | NYP'da Özel Konular: Abstract Classes, Interfaces, Enum Sınıfları |
| 7 | 12.04.2022 | Exception Handling, Unit Test |
| 8 | 21.04.2022 | 1. Ara Sınav (10:00-12:00) |
| 9 | 26.04.2022 | Temel Veri Yapılarının Jenerik Sınıflar Eşliğinde Kullanımı (Liste ve Eşleme Yapıları). |
| 10 | 03.05.2022 | Ramazan Bayramı |
| 11 | 10.05.2022 | Tip dönüşümü, Dosyalar ve Akışlar ile Çalışmak (Serileştirme ve Ters İşlemi), İç Sınıflar |
| 12 | 17.05.2022 | Paralel Programlamaya Giriş - Multithreading |
| 13 | 24.05.2022 | 2. Ara Sınav |
| 14 | 31.05.2022 | GUI (Graphical User Interface) Kavramlarına Giriş |

Öğr. Grv. Furkan ÇAKMAK

# MULTITHREADING

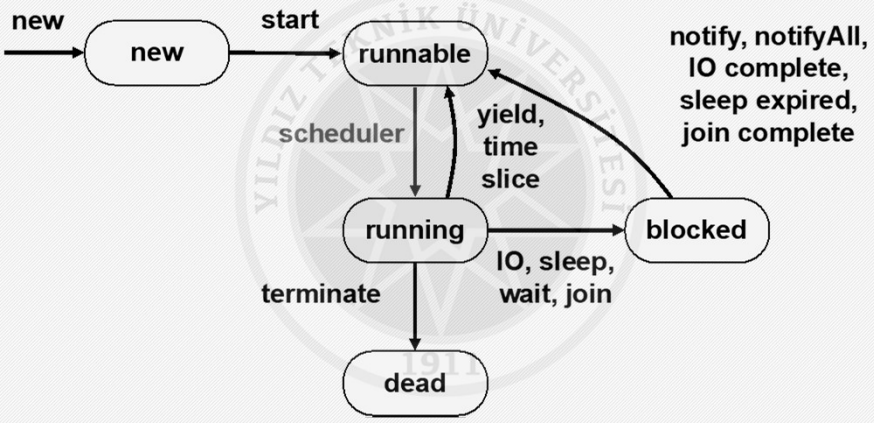


- Multitasking, multiple processes and multithreading:
  - Multitasking is the ability to have more than one program working at the same time.
  - Nowadays, you are likely to have a computer with its CPU having multiple cores.
  - Each core can execute one or more tasks, i.e. processes, depending on the CPU architecture.
  - A process can sometimes be divided into threads that may run in parallel, i.e. concurrently running sub-processes.
    - If there are enough hardware resources, i.e. cores, the time it takes to complete a process will drop significantly.
    - However, this increase in the performance will not be in the order of the available cores.
      - The concurrently running threads will sooner or later need to synchronize with each other.
      - Moreover, creating a process or a thread takes some execution time as well.



# MULTITHREADING (CON'T)





# MULTITHREADING (CON'T)



- How should a thread wait?
  - If a thread is unable to continue its task because of an obstacle, that thread should wait until the obstacle has been removed.
    - Obstacle: The needed information has not arrived from: the network, another thread, the user, etc.
  - You should not do "busy waiting", i.e. executing dummy instructions such as running empty loops for 10.000 times.
  - Instead, you should put that thread into the blocked state by using the sleep command.
  - A sleeping thread, unlike a busy waiting one, does not consume system resources.
  - A sleeping thread is at risk of becoming unable to awake.
    - You must catch the java.lang.InterruptedException, which is a checked exception.



# MULTITHREADING (CON'T)



- Procedure for running a task in a separate thread:
  1. Place the code for the task into the run method of a class that implements the Runnable interface.
  2. Create an object of your class
  3. Create a Thread object from the Runnable
  4. Start the thread by using Thread.start method (do not call the run method directly)
- Do not code your own threads by inheriting from the Thread class.
  - Otherwise you will lay your only inheritance right to waste.
- Let's make a demonstration with a nonsense application about people watching a match:
  - Each person will shout for the team they support when he or she becomes excited.
  - There is a possibility for each person to become excited in 0-1000 ms.
  - Each person become exhausted after shouting 10 times.

# MULTITHREADING (CON'T)



```
package nyp14a;
import java.util.Random;

public class SoccerFan implements Runnable {
     public final static int STEPS = 10;
     public final static int DELAY = 1000;
     private String teamName, shoutPhrase;

     public SoccerFan( String teamName, String shoutPhrase ) {
          this.teamName = teamName;
          this.shoutPhrase = shoutPhrase;
     }

     public void run() {
          Random generator = new Random();
          try {
              for( int i = 0; i < STEPS; i++ ) {
                    System.out.println( teamName + " " + shoutPhrase );
                    Thread.sleep( generator.nextInt(DELAY) );
              }
          }
          catch (InterruptedException e) {
              e.printStackTrace();
          }
     }
}
```



# MULTITHREADING (CON'T)

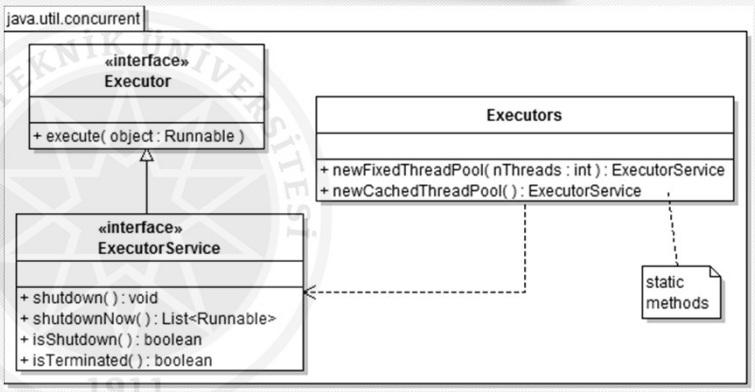


```
package nyp14a;

public class Match {
     public static void main(String[] args) {
          Thread aThread;
          aThread = new Thread( new SoccerFan("G.S.", "Rulez!") );
          aThread.start( );
          aThread = new Thread( new SoccerFan("G.S.", "is the champ!") );
          aThread.start( );
          aThread = new Thread( new SoccerFan("F.B.", "is no.1!") );
          aThread.start( );
          aThread = new Thread( new SoccerFan("F.B.", "is the best!") );
          aThread.start( );
     }

}
```

# MULTITHREADING (CON'T)



- Thread pools:
  - Running a small number of tasks in separate threads is acceptable.
  - But do not forget that actual processing units in a typical CPU is rather low, and creating a thread has also a processing cost.
  - Therefore, if you are to execute a large number of tasks, you should use a thread pool instead.
  - Java provides the following interfaces and classes for this purpose:






# MULTITHREADING (CON'T)

BLM2012 Nesneye Yönelik Programlama Hafta 12

- java.util.concurrent.ExecutorService:
  - public void shutdown( ) :
    - Shuts down the executor, but allows the tasks currently in the pool to be completed. New threads are not accepted to the pool.
    - We need to use this method for a safe ending.
  - public List<Runnable> shutdownNow( )
    - Shuts down immediately, stops the unfinished threads and returns them in a list.
  - public boolean isShutdown( ):
    - Returns true if the executor is shut down.
  - public boolean isTerminated( ):
    - Returns true if all the tasks in the pool are terminated.
    - Can be used in the main method for waiting the threads to be finished

# MULTITHREADING (CON'T)



- java.util.concurrent.Executor:
  - public void execute( Runnable object ): Executes the given task
- java.util.concurrent.Executors:
  - public static ExecutorService newFixedThreadPool( nThreads : int )
    - Creates a thread pool that reuses a fixed number of threads
  - public static ExecutorService newCachedThreadPool( )
    - Creates a thread pool that creates new threads as needed, but will reuse previously constructed threads when they are available



# MULTITHREADING (CON'T)

BLM2012
Nesneye
Yönelik
Programlama
Hafta 12

- IDE'den gösterim yapalım.

# MULTITHREADING (CON'T)



- Exceptions and multithreading:
  - Throwing an unchecked exception from the run() method is easy.
  - You cannot change the method signature of the run method to declare that an unchecked exception can be thrown.
  - To throw a checked exception from the run() method, you need to:
    1. Code the multithreaded task that can throw the exception in a normal member method
    2. Declare that method as throws SomeCheckedException
    3. Call that method from run() and use try/catch properly.



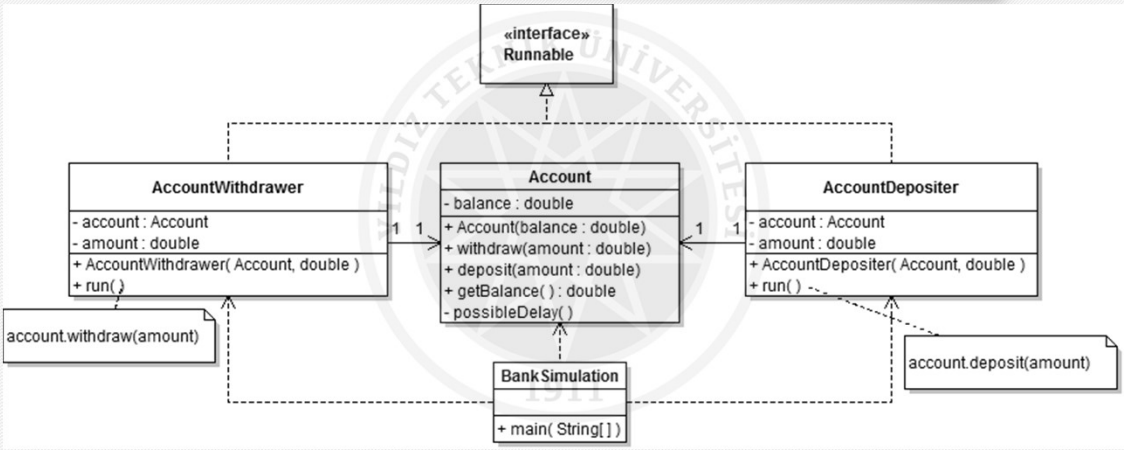
# MULTITHREADING (CON'T)



- Race condition:
  - In most practical multithreaded applications, two or more threads need to share access to the same data.
  - What happens if two threads have access to the same object and each calls a method that modifies the state of the object?
    - As you might imagine, the threads can step on each other's toes!
    - Depending on the order in which the data were accessed, corrupted objects can result.
    - Such a situation is often called a race condition.

# MULTITHREADING (CON'T)







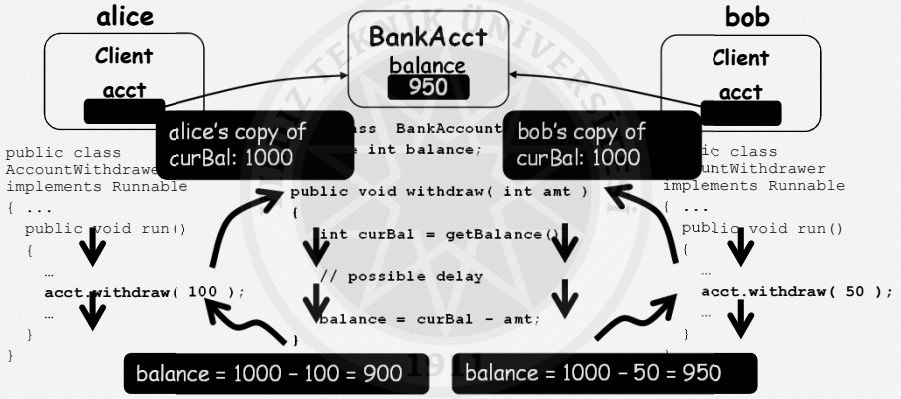

# MULTITHREADING (CON'T)



- About the data structures and multithreading:
  - Remember the data structures section: Some data structures are thread-safe, i.e. synchronized
  - Vector<E> and Hashtable<K,V>
  - Use those data structures when multithreading is to be used.
- IDE'den gösterim yapalım.

# MULTITHREADING (CON'T)








# Sabırla Dinlediğiniz İçin Teşekkürler