Synchronization

There are situations where we do NOT want methods (or code) to be invoked in parallel (multi-threaded) but rather sequentially.

Example: a method print() of printer should be used sequentially (> printer queue) and not in parallel).

StartThreads.java StartThreads2.java Printer.java

```
1  package threads;
2
3  public class Printer {
4      public synchronized void print() {
5          System.out.println("Printer start printing");
6
7          //method access the printer hardware
8
9          System.out.println("Printer ends printing");
10     }
11 }
12
```

Option 1 :

Synchronizing an entire method

StartThreads.java StartThreads2.java Printer.java

```
1  package threads;
2
3  public class Printer {
4      public void print() {
5          System.out.println("Printer start printing");
6
7          synchronized(new Object()) {
8              //method access the printer hardware
9          }
10
11         System.out.println("Printer ends printing");
12     }
13 }
14
```

Option 2 :

Synchronizing just a code block (i.e. a part of a method)

Printer.java

```
1  package threads;
2
3  import java.util.concurrent.locks.Lock;
4  import java.util.concurrent.locks.ReentrantLock;
5
6  public class Printer {
7      public void print() {
8          System.out.println("Printer start printing");
9
10             Lock lock = new ReentrantLock();
11             lock.lock();
12
13             //method access the printer hardware
14
15             lock.unlock();
16
17         System.out.println("Printer ends printing");
18     }
19 }
20
```

Option 3:

```java
package threads;

public class Bank {
    private int[] accounts = {30, 50, 100};

    public synchronized void transferMoney(int fromAccount, int toAccount, int amount) {

        int oldAmount = accounts[fromAccount];
        int newAmount = oldAmount - amount;
        accounts[fromAccount] = newAmount;

        try {
            //Thread.sleep(5000);
            Thread.sleep((int)Math.random() * 10000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }


        int oldAmountTo = accounts[toAccount];
        int newAmountTo = oldAmountTo + amount;
        accounts[toAccount] = newAmountTo;
    }

    public synchronized void printBalance(){
        System.out.println("\nThe balances are ");
        for (int i = 0; i < accounts.length; i++) {
            System.out.println("The balance of account " + i + " is "+ accounts[i]);
        }
    }
}
```

synchronized

synchronized

```java
package threads;

public class Clerk extends Thread {
    private Bank bank;
    private int fromAccount;
    private int toAccount;
    private int amount;



    public Clerk(Bank bank, int fromAccount, int toAccount, int amount) {
        super();
        this.bank = bank;
        this.fromAccount = fromAccount;
        this.toAccount = toAccount;
        this.amount = amount;
    }

    public void run() {
        bank.transferMoney(fromAccount, toAccount, amount);
        bank.printBalance();
    }
```

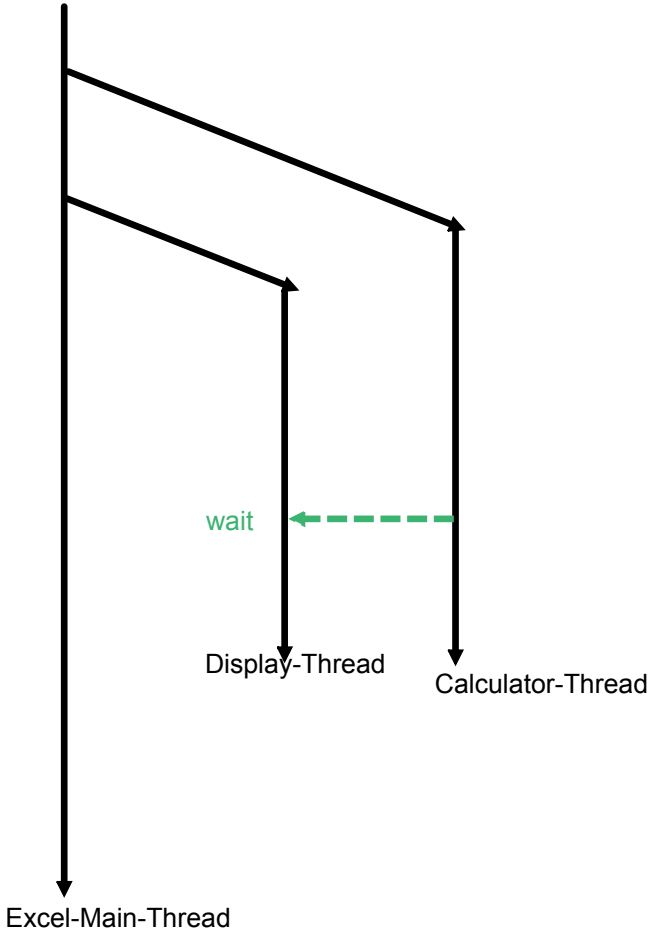
```java
package threads;

public class StartBank {
    public static void main(String[] args) {

        Bank bank = new Bank();
        bank.printBalance();

        Clerk clerk1 = new Clerk(bank, 0, 1, 20);
        Clerk clerk2 = new Clerk(bank, 1, 2, 20);
        Clerk clerk3 = new Clerk(bank, 2, 0, 20);

        clerk1.start();
        clerk2.start();
        clerk3.start();
    }
}
```


wait
Display-Thread
Calculator-Thread
Excel-Main-Thread

# Regular Expressions

Classes

. (dot) any character

\w word characters / letters A-Z a-z 0-9

\W non-word characters

\s white space characters empty, CR, TAB, LF

\S non-white-space characters

\d digits [0-9]

\D non-digits

[ ] Own class e.g. [aeiou] [A-Za-z]

() Alternatives e.g. (de|com|org)

Special characters

* Previous pattern repeated 0 to n times

+ Previous pattern repeated 1 to n times

? Previous pattern repeated 0 or 1 time

{n,m} Previous pattern repeated min n and max m times

{n,} Previous pattern repeated min n times

{,m} Previous pattern repeated max m times

\ Escape Character

Working with JUnit

1. File > new > Source Folder (name of new folder "test")
2. File > new > JUnit Test Case (name of class = name of class under tests + "Test" (e.g. EmailCheckTest"))
3. Implement test code (in our case just copy from assignment)

Assignment: implement the method "checkValidAdress"

Offer: submit 10 exercises for final exam