DIAGNOSTIC TEST
Software Systems

Date: Week 7

- This diagnostic test contains questions on exceptions and concurrency
- Answer the questions as well as you can **in 30 minutes**
- Discuss your solutions in the next **10 minutes** with your neighbour.
- The correct answers will be presented during the last **10 minutes** of the meeting.

Question 1 (Exceptions)

Regard the following class `BankAccount`.

```
public class BankAccount {
    private final String name;
    private double balance;

    public BankAccount(String name, double balance) {
        this.name = name;
        if (balance > 0) {
            this.balance = balance;
        } else {
            this.balance = 0;
        }
    }

    public String getName() {
        return this.name;
    }

    public boolean deposit(double amount) {
        if (amount < 0) {
            return false;
        }
        this.balance = this.balance + amount;
        return true;
    }

    public boolean withdraw(double amount) {
        if (amount > this.balance || amount < 0) {
            return false;
        }
        this.balance = this.balance - amount;
        return true;
    }
}
```

Answer the following questions:

a. Adapt `BankAccount` such that it throws *disctinct* exceptions in case an attempt is made to (i) create an account with a negative initial balance; (ii) transfer a negative amount; and (iii) withdraw a negative amount or an amount larger than the current balance. Make sure your solution satisfies the following conditions:

- The thrown exceptions should not be standard Java exceptions, i.e., they should all be user-defined;
- The return types of `deposit` and `withdraw` should be modified;
- Your exception classes should have `getMessage()` methods that return sensible error messages;
- At least one of your exception classes should store the illegal (negative) amount in a field of the class.

b. Write a sequence of instructions that causes each of the three exceptions to be thrown, and then catches them and prints the error message on the standard output. None of your declared exceptions should be left uncaught.

Question 2 (Concurrency)

Consider the following class `Cell`

```
public class Cell {
    int v = 1;

    synchronized public int get() {
        return this.v;
    }

    synchronized public void set(int v) {
        this.v = v;
    }
}
```

and the classes `Worker`, with subclasses `Alex` en `Brenda`:

```
public class Worker extends Thread {
    protected Cell c;

    public Worker(Cell c) {
        this.c = c;
    }
}
```

```
public class Alex extends Worker {
    public Alex(Cell c) {
        super(c);
    }

    public void run() {
        int v = this.c.get();
        this.c.set(v + 4);
    }
}
```

```java
public class Brenda extends Worker {
    public Brenda(Cell c) {
        super(c);
    }

    public void run() {
        int v = this.c.get();
        this.c.set(v * 4);
    }
}
```

Now consider the following code using these classes:

```java
1  public class Usage {
2      public static void main(String[] args) {
3          Cell cell = new Cell();
4          Worker w1 = new Alex(cell);
5          Worker w2 = new Brenda(cell);
6          w1.start();
7          w2.start();
8          try {
9              w1.join();
10             w2.join();
11         } catch (InterruptedException e) {
12         }
13         System.out.println(cell.get());
14     }
15 }
```

Answer the following questions, including explanations of the answer.

a. Which values can `Usage.main` print to the standard output?

b. If you remove the entire `try-catch`-block with the `join`-calls (lines 8–12) from `main`, which values can `main` then print to the standard output?

c. If you remove the `try-catch`-block (as above) and in addition change the calls `w1.start()` and `w2.start()` (lines 6 and 7) into `w1.run()` and `w2.run()`, which values can `main` then print to the standard output?

d. Start with the original `main`-method (without the changes proposed above) and change the method `set` of class `Cell` into:

```java
public synchronized void set(int v) {
    this.v = v;
    notifyAll();
}
```

Which values can `main` now print to the standard output?

e. Start with the original classes and change `run` in class `Alex` as follows:

```java
public void run() {
    synchronized(c) {
        c.set(c.get()+4);
    }
}
```

and `run` in `Brenda` as follows:

```
public void run() {
    synchronized(c) {
        c.set(c.get()*4);
    }
}
```

Which values can `main` now print to the standard output?

f. Start with the original classes and define in `Cell`

```
public static Object o1 = new Object();
public static Object o2 = new Object();
```

Change the method `run` of class `Alex` into

```
public void run() {
    synchronized(o1) {
        synchronized(o2) {
            int v=c.get();
            c.set(v+4);
        }
    }
}
```

and the method `run` of class `Brenda` into

```
public void run() {
    synchronized(o2) {
        synchronized(o1) {
            int v=c.get();
            c.set(v*4);
        }
    }
}
```

Which values can `main` now print to the standard output?