Instuderingsuppgifter läsvecka 7

1.

Betrakta klassen Product nedan:

```
public class Product {
    private int nrInStock = 0;
    private int outwardConnectionCount;
    public void addToStock(int quantity) {
        nrInStock = nrInStock + quantity;
    }// addToStock
    public void removeFromStock(int quantity) {
       if (quantity > nrInStock)
           throw new IllegalArgumentExecption("To big quantity");
        nrInStock = nrInStock - quantity;
    }// addToStock
    public int numberInStock() {
       return  nrInStock;
    }// numberInStock
    //fler attribut och metoder som är oväsentliga för uppgiften
}// Product
```

Klassen Product är inte trådsäker (*thread-safe).*

a) Förklara vad som menas med att en klass inte är trådsäker. Beskriv med ett exempel vad som kan inträffa i klassen Product.

b) Gör klassen Product trådsäker. Din lösning skall så längt som möjligt bibehålla parallella bearbetning.

2.

När man skriver ett program i Java som består av flera samverkande trådar uppstår ofta ett behov av att synkronisera trådarnas exekvering. Till exempel kan en tråd behöva vänta på att en annan tråd ska bli färdig med någon beräkning eller så behöver man se till att inte två trådar försöker använda sig av samma resurs samtidigt. En ofta använd teknik i Java är att använda sig av så kallade monitorer. För att stödja användandet av monitorer har man i Java infört ett antal språkkonstruktioner samt ett antal standardmetoder i klassen Object.

a) Beskriv vad följande standardmetoder har för innebörd (semantik) i Java:

- wait()
- notify()
- notifyAll()

b) För att man ska kunna anropa metoderna wait(), notify() eller notifyAll() (utan att få en exception), måste ett mycket specifikt villkor vara uppfyllt. Vilket?

3.

Kalle Klåpare har skrivit ett Javaprogram som använder två trådar, thread1 och thread2. Nu vill han att thread1 ska kunna skicka över en sträng till thread2. Han har därför skrivit följande klass för att hantera överföringen:

```
public class Buffer {
   private String message;
   public void set(String m) {
      message = m;
   }//set
   public String get() {
      String m = message;
      message = null;
      return m;
   }//get
}//Buffer
```

thread1 anropar då och då set() för att lägga in en sträng i bufferten. thread2 anropar hela tiden get() och när något annat än **null** returneras utför thread2 någon typ av operation på den mottagna strängen. Tyvärr fungerar Kalles program väldigt dåligt.

a) Ge exempel på två problem Kalle kan råka ut för med lösningen ovan.

b) Skriv om klassen Buffer ovan så att den blir både trådsäker och effektiv.

4.

Betrakta nedanstående program:

```
public class Red {
   public static void main(String[] args) throws InterruptedException {
      Thread t1 = new Thread(new Runnable() {
         public void run() {
            System.out.print("O");
            System.out.print("Y");
         }
      });
      Thread t2 = new Thread(new Blue());
      System.out.print("R");
      t1.start();
      System.out.print("G");
      t2.start();
      System.out.print("I");
      t1.join();
      System.out.print("V");
      t2.join();
      System.out.print("K");
   }
}//Red

public class Blue implements Runnable {
   public void run() {
      System.out.print("B");
   }
}//Blue
```

Vilka av nedanstående utskriftssekvenser är möjliga/omöjliga att erhålla från programmet?

a) ROYGBIVK b) ROYBGIVK c) RGOYIBVK d) OYBRGIVK