Tentamen i Objektorienterad modellering och diskreta strukturer

Lösningar

1. **a.**

```
public abstract class Member {
    protected String name;
    public abstract int amount();
        public String payment() {
            return name + "den avgift du ska betala är" + amount();
        }
    }

public class Senior extends Member {
    public int amount() {
        return 3000;
    }
}
```

Klasserna Junior och Passive analoga med Senior.

b.

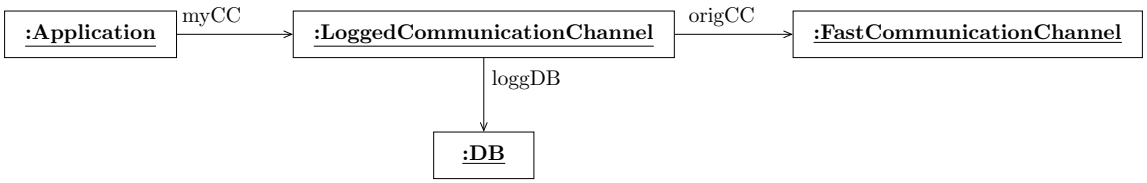
```
abstract public class Member {
    private String name;
    TypeOfMembership membership;

    public Member(String namn, TypeOfMembership medlemskap) {
        this.name = namn;
        this.membership = medlemskap;
    }

    public void setMedlemskap(TypeOfMembership ms) {
        membership = ms;
    }

    public String payment() {
        return name + "den avgift du ska betala är" +
            membership.amount();
    }
}

public interface TypeOfMembership {
    int amount();
}

public class SeniorStrategy implements TypeOfMembership {
    public int amount() {
        return 3000;
    }
}
```

2.
```
public class Account extends Observable {
    protected int saldo;
    public void deposit(int amount) {
        saldo += amount;
        setChanged();
        notifyObservers();
    }
    public void withdraw(int amount) {
        saldo -= amount;
        setChanged();
        notifyObservers();
    }
    public int getSaldo() {
        return saldo;
    }
}

public class Alarm implements Observer {
    protected int limit;
    AccountManager theAccount;
    public Alarm(AccountManager theAccount, int limit) {
        this.limit = limit;
        this.theAccount = theAccount;
        theAccount.addObserver(this);
    }

    @Override
    public void update(Observable o, Object arg) {
        if (theAccount.getSaldo() < limit) {
            alert();
        }
    }

    public void alert() {
        // omissions
    }
}
```

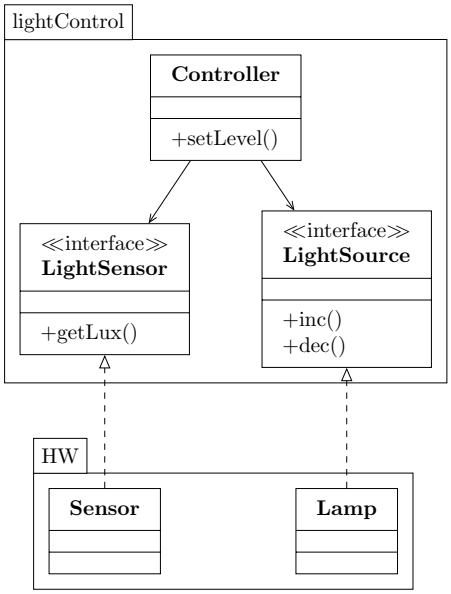
3.
```
public class Application {
    CommunicationChannel myCC = new FastCommunicationChannel();
    public static void main(String[] args) {
        // omissions
    }
    private void send(String text) {
        myCC.send(text);
    }
    private String receive() {
        return myCC.receive();
    }
    public void startLogg(DB loggDB) {
        myCC = new LoggedCommunicationChannel(loggDB, myCC);
    }
    public void stopLogg() {
        if (myCC instanceof LoggedCommunicationChannel) {
            myCC = ((LoggedCommunicationChannel)myCC).getOrigCC();
        }
    }
}
public interface CommunicationChannel {
    public void send(String text);
    public String receive();
}
public class LoggedCommunicationChannel implements CommunicationChannel {
    CommunicationChannel origCC;
    DB loggDB;
    public LoggedCommunicationChannel(DB db, CommunicationChannel origCC) {
        this.origCC = origCC;
        this.loggDB = db;
    }
    public CommunicationChannel getOrigCC() {
        return origCC;
    }
```

```
    public void send(String text) {
        loggDB.appendSent(text);
        origCC.send(text);
    }
    public String receive() {
        String text;
        text = origCC.receive();
        loggDB.appendReceived(text);
        return text;
    }
}
```

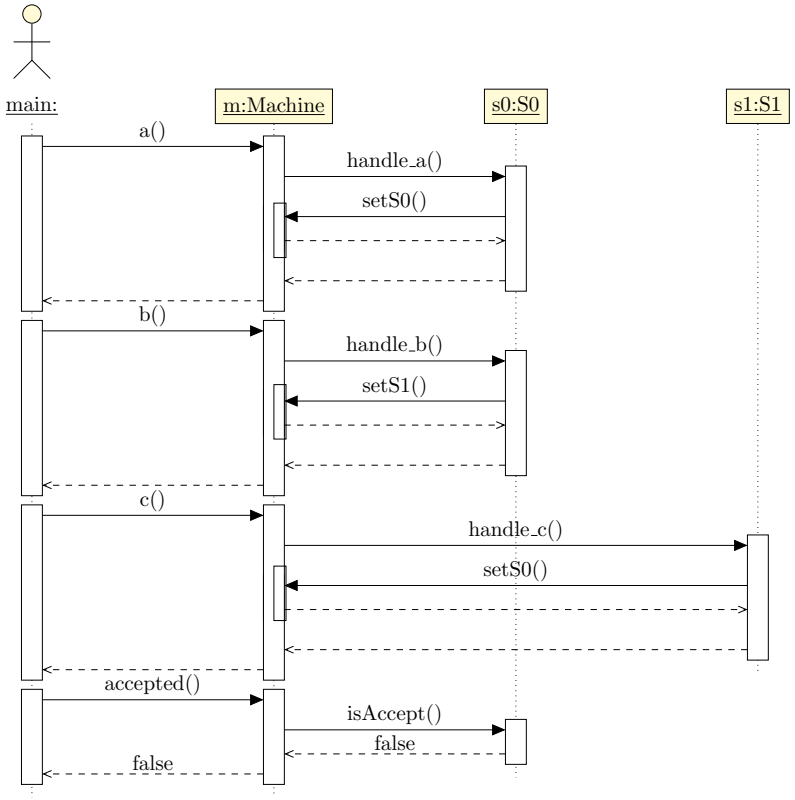
Objektdiagram (kallades instansmodell i uppgiften):



4. Klassdiagram:



5.



6. **a**. $a^*b(ca^*b)^*$ eller, med utökad syntax, $(a|bc)^*b$

b. ('aaab', 'abcb', 'bcab')

7. Härledningen som beviskräd respektive "tableau":

$$\dfrac{\dfrac{\neg(P \wedge Q) \qquad \dfrac{[\cancel{Q}] \quad [\cancel{P}]}{P \wedge Q}\,[\wedge_I]}{\neg P}\,[\neg_I]}{Q \to \neg P}\,{\to_I}$$

1	$\neg(P \wedge Q)$	P
2	Q	
3	$\quad P$	
4	$\quad P \wedge Q$	$\wedge_I$, 2, 3
5	$\quad \mathsf{F}$	$\wedge_I$, 1, 4
6	$\quad \neg P$	$\neg_I$ (RAA), 1, 5
7	$Q \to \neg P$	$\to_I$, 2, 6

8. **a**. $\{(x, y) \in \mathbb{N} \times \mathbb{N} \mid \exists z \in \mathbb{N} \,.\, x < z \wedge z > y\}$

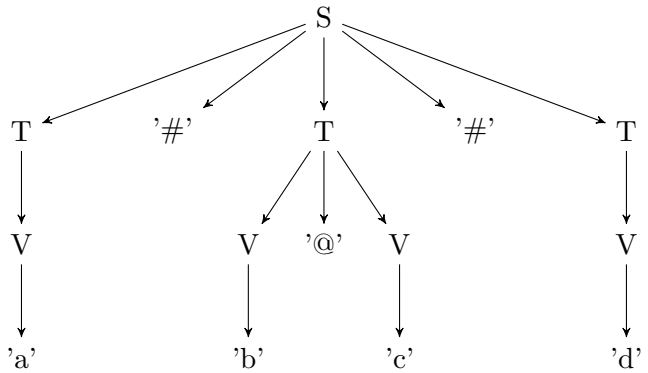
b. $\{(0,0), (0,1), (1,0), (1,1)\}$

Lösningen uttryckt i haskell: Definiera en funktion, `ltgt`, som ger en lista med elementen som ingår i relationen $(<);(>)$ på mängden m. Svaret till b) ges av att anropa `ltgt` på mängden $\mathbb{N}_3$:

```
ltgt m =  [(x,y) | x <- m, y <- m, z <- m, x < z && z > y]
ltgt3 = ltgt [0,1,2]
```

NB, `ltgt` ger en lista som kan innehålla dubletter. Dessa kan tas bort med funktionen `nub`.

9. Syntaxträdet kan representeras:



Man kan även tolka grammatiken så att det syns i trädet att operatorerna `#` och `@` är binära, genom att skriva om upprepningarna (`X*`) i grammatiken till BNF med rekursiva regler:

```
S := T | S '#' T
T := V | T '@' V
V := LETTER
```

Då blir syntaxträdet