Tentamen i Objektorienterad modellering och diskreta strukturer

Lösningar

1.
```
public class Add implements Instruction {
  private int address1, address2, address3;

  public void execute(Word [] memory) {
      memory[address3].add(memory[address1],
                           memory[address2]);
    }
  }
}
```

```
public class Word {
  private int value;
  public void add(Word word1, Word word2) {
    value = word1.value + word2.value;
  }

  public String toString() {
    return String.valueOf(value);
  }
}
```

2.
```
public abstract class Event {
  protected int time;
  public abstract void execute(Context context);
  public abstract String kind();
  public String toString() {
    return kind() + " " + time;
  }
}
```

```
public class Arrival extends Event {
  public void execute(Context context) {
      context.arrival(time);
  }
  public String kind() {
    return "ARRIVAL";
  }
}
```

Departure och Measurement är analoga.

3.
```
public class Account {
  private Revenue revenue;
  private float balance;
  public void setRevenue(Revenue revenue) {
    this.revenue = revenue;
  }
  public float revenue(int days) {
    return revenue.amount(days, balance);
  }
}
```

```
public interface Revenue {
  public float amount(int days, float balance);
}
public class Standard implements Revenue {
  public float amount(int days, float balance) {
    return balance*days*4.0/365;
  }
}
```

4.
```
public interface BinaryTree {
    public void printNodes();
}

public class Empty implements BinaryTree {
  public void printNodes() {
  }
}
```

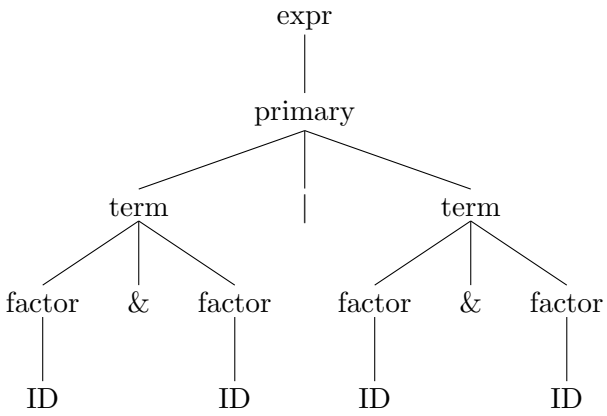
```
public class Node implements BinaryTree {
  private Object data;
  private BinaryTree left, right;

  public Node(Object data, BinaryTree left,
                           BinaryTree right) {
    this.data = data;
    this.left = left;
    this.right = right;
  }
  public void printNodes() {
    left.printNodes();
    System.out.println(data);
    right.printNodes();
  }
}
```

5. **a**

```
public Expr substitute(Variable variable1, Term term) {
    if (variable.equals(variable1)) {
        return this;
    } else if (term.contains(variable)) {
        Variable newVariable = new Variable();
        Expr newExpr = expr.substitute(variable, newVariable);
        return new ForAll(newVariable, newExpr.substitute(variable1, term));
    } else {
        return new ForAll(variable, expr.substitute(variable1, term));
    }
}
```

b $\rho^0 = \{(0,0),(1,1),(2,2)\}$, $\rho^2 = \{(0,1),(1,1),(0,2),(1,2)\}$ och
$\rho^* = \{(0,0),(0,1),(0,2),(1,1),(1,2),(2,2)\}$.

6. **a**



b

```
expr      ::= term ('->' term)?
term      ::= factor (('|' | '&') factor)*
factor    ::= ID | '!' factor | '(' expr ')'
```