Tentamen i Objektorienterad modellering och design

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.
```
package model;
import expr.Expr;
import expr.Environment;
public class Sheet implements Environment {
    private Map<Address, Slot> map = new HashMap<Address, Slot>();
    public double value(Object key) {
        Slot slot = map.get(key);
    // omissions
    }
}

public interface Slot {
    public double value(Environment environment);
}


package expr;
public interface Environment {
  public double value(Object key);
}
class AddressExpr extends Expr {
    private Address address;
    public double value(Environment environment) {
        return environment.value(address);
    }
}
```