Tentamen i Objektorienterad modellering och design

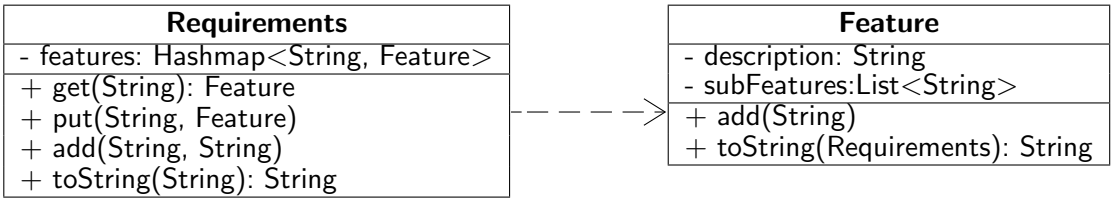
Lösningar

1.
```
public interface Word {
  public void add(Word word1, Word word2);
}
public class Add implements Instruction {
  private Word word1, word2, word3;
  public void execute() {
    word3.add(word1, word2);
  }
}
public class LongWord implements Word{
  private long value;
  public void  add(Word word1, Word word2) {
    value = ((LongWord)word1).value + ((LongWord)word2).value;
  }
}
```

2.
```
public interface Expr {
  public int value();
}
public interface Operation {
  public int value(int op1, int op2);
}

public class Int implements Expr {
  private int value;
  public int value() {
    return value;
  }
}
public class BinaryExpr extends Expr {
  private Expr expr1, expr2;
  private Operation operation;

  public setOperation(Operation operation) {
    this.operation = operation;
  }

  public int value() {
    return operation(expr1.value(), expr2.value())
  }
}

public class Plus implements Operation {
  public int value(int op1, int op2) {
    return op1 + op2;
  }
}

Klassen Times är analog

new BinaryExpr(new BinaryExpr(new Int(1), new Int(2), new Plus()),
    new Int(3), new Times())
```

3.
```
public abstract class FileMenuItem extends JMenuItem implements ActionListener {
    protected XL xl;
    protected Sheet sheet;

    protected FileMenuItem(XL xl, Sheet sheet, String title) {
        super(title);
        this.xl = xl;
        this.sheet = sheet;
        addActionListener(this);
    }

    protected abstract void action(File file);
    protected abstract int openDialog(JFileChooser fileChooser);

    public void actionPerformed(ActionEvent event) {
        JFileChooser fileChooser = new JFileChooser(".");
        int option = openDialog(fileChooser);
        if (option == JFileChooser.APPROVE_OPTION) {
            File file = fileChooser.getSelectedFile();
            action(file);
        }
    }
}
public class LoadMenuItem extends FileMenuItem {

    public LoadMenuItem(XL xl, Sheet sheet) {
        super(xl, sheet, "Load");
    }

    protected void action(File file) {
        sheet.load(new XLBufferedReader(file));
    }

    protected int openDialog(JFileChooser fileChooser) {
        return fileChooser.showOpenDialog(xl);
    }
}

Klassen SaveMenuItem är analog
```

4. **a**.



b.
```
I Requirements:

  public toString(String name) {
    return features.get(name).toString(this);
  }

  I Feature:

  public String toString(Requirements requirements) {
    StringBuilder builder = new StringBuilder();
    builder.append(description);
```

```
        for(String subFeature: subFeatures) {
           builder.append(requirements.get(subFeature).toString(requirements));
        }
        return builder.toString();
      }
```

5.
```
class ErrorFeature extends Feature {
    public String toString(Requirements requirements) {
        throw new CircularException();
    }
}

I Requirements:

  public void add(String name, String addedName) {
    Feature feature = features.get(name);
    feature.add(addedName);
    features.put(name, new ErrorFeature());
    feature.toString(this);
    features.put(name, feature);
  }
```