Anhang zum theoretischen Teil des Leistungsnachweises

```java
public class Tool {
    private static int nr = 1;
    private static final double DELTA = 16;
    public static Intervall gibMir(int i) {
        final double oben = 15 * i;
        return new Intervall(oben - DELTA, oben);
    }
    public static Intervall gibMir() {
        return gibMir(nr++);
    }
}
```

```java
public interface Pool {
    Pool clone();
    Object gibMir();
    void nimmDir(int a);
}
```

```java
public enum CoolBool {
    TRUE, FALSE, WHO_CARES
}
```

```java
public class ToolPool implements Pool {
    private int nr = 1;
    @Override
    public Object gibMir() {
        return Tool.gibMir(nr++);
    }
    @Override
    public Pool clone() {
        try {
            return (Pool) super.clone();
        } catch(CloneNotSupportedException e) {
            return this;
        }
    }
    @Override
    public void nimmDir(int anzahl) {
        for (int i = 0; i < anzahl; i++) gibMir();
    }
    @Override
    public int hashCode() {
        return nr;
    }
    @Override
    public boolean equals(Object o) {
        if (o == this) return true;
        if (o == null) return false;
        if (getClass() != o.getClass()) return false;
        ToolPool tp = (ToolPool) o;
        if (nr != tp.nr) return false;
        return true;
    }
}
```

```java
public class CoolPool extends ToolPool implements Cloneable {
    public Intervall gibMir() {
        return (Intervall) super.gibMir();
    }
}
```

```java
public class BoolPool extends CoolPool {
    private CoolBool bool = CoolBool.WHO_CARES;
    public boolean equals(Object o) {
        if (!super.equals(o)) return false;
        BoolPool bp = (BoolPool) o;
        if (bool != bp.bool) return false;
        return true;
    }
    public Intervall gibMir() {
        if (bool != CoolBool.TRUE) super.gibMir();
        bool = CoolBool.TRUE;
        return super.gibMir();
    }
    public void nimmDir(int a) {
        if (bool == CoolBool.TRUE) super.nimmDir(a);
        bool = CoolBool.FALSE;
        super.nimmDir(a);
    }
}
```

```java
public class FoolPool extends ToolPool {
    public Pool clone() {
        return new ToolPool();
    }
    public String gibMir() {
        return "Frieden";
    }
    public boolean equals(Object o) {
        return true;
    }
}
```

```java
public class PoolPool extends ToolPool implements Cloneable {
    private Pool pool = new ToolPool();
    public PoolPool(Pool p) {
        pool = p;
    }
    @SuppressWarnings("unchecked")
    public Pool clone() {
        PoolPool kopie = (PoolPool) super.clone();
        if (!(pool instanceof BoolPool)) kopie.pool = pool.clone();
        return kopie;
    }
    public boolean equals(Object o) {
        if (!super.equals(o)) return false;
        if (getClass() != o.getClass()) return false;
        PoolPool pp = (PoolPool) o;
        if (!pp.pool.equals(pool)) return false;
        return true;
    }
}
```