



Вариант № 86006  
Лабораторная работа № 3  
по дисциплине  
*'Программирование'*

Выполнил:  
Студент группы Р3113  
*Куперштейн Дмитрий; : 269359*  
Преподаватель:  
*ПИСЬМАК АЛЕКСЕЙ ЕВГЕНЬЕВИЧ*

## Содержание

1	Задание	3
2	Диаграмма классов объектной модели	4
3	Исходный код программы	5
4	Результат работы программы	10
5	Вывод	10

## 1 Задание

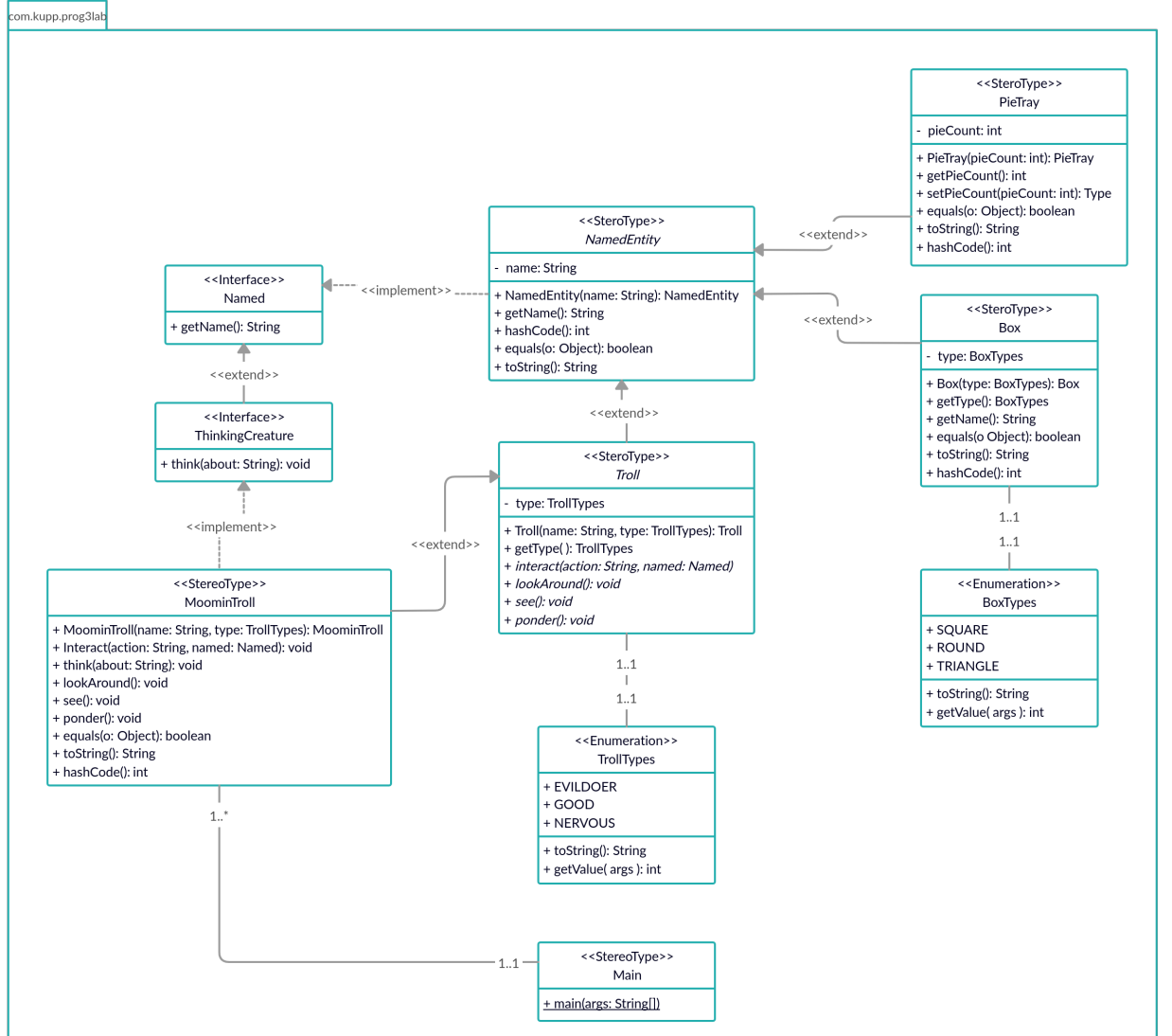
Описание предметной области, по которой должна быть построена объектная модель (для варианта 86006):

"Страсть как забавно, – подумала она. – Вот уж удивится моя сестра!" Оглядевшись вокруг, она заметила плывшие рядом поднос для пирожков и шкатулку Муми-мамы. После недолгого раздумья (хотя на подносе еще оставалось несколько пирожков) она выбрала шкатулку и залезла туда.

Программа должна удовлетворять следующим требованиям:

1. Доработанная модель должна соответствовать принципам SOLID.
2. Программа должна содержать как минимум два интерфейса и один абстрактный класс (номенклатура должна быть согласована с преподавателем).
3. В разработанных классах должны быть переопределены методы `equals()`, `toString()` и `hashCode()`.

## 2 Диаграмма классов объектной модели



## 3 Исходный код программы

### 3.0.1 Main.java

```
package com.kupp.prog3lab;

public class Main {
    public static void main(String[] args) {
        MoominTroll she = new MoominTroll("She", TrollTypes.GOOD);
        PieTray tray = new PieTray(8);
        Box box = new Box(BoxTypes.ROUND);
        she.think("that's funny");
        she.think("Sister would be surprised");
        she.lookAround();
        she.see(tray);
        she.see(box);
        she.ponder();
        int pieCount = tray.getPieCount();
        if (pieCount > 2) {
            she.interact(String.format("ate %d pies off", pieCount - 2), tray);
            tray.setPieCount(2);
        }
        she.interact("climbed into", box);
    }
}
```

### 3.0.2 Named.java

```
package com.kupp.prog3lab;

public interface Named {
    String getName();
}
```

### 3.0.3 NamedEntity.java

```
package com.kupp.prog3lab;

public abstract class NamedEntity implements Named {
    private String name;

    public NamedEntity(String name) {
        this.name = name;
    }

    public String getName() {
        return this.name;
    }

    @Override
    public int hashCode() {
        return getName().hashCode();
    }

    @Override
    public boolean equals(Object o) {
```

```

        if (!(o instanceof NamedEntity)) {
            return false;
        }
        NamedEntity other = (NamedEntity) o;
        return other.getName().equals(this.getName());
    }

    @Override
    public String toString() {
        return String.format("NamedEntity with name \"%s\"", this.getName());
    }
}

```

### 3.0.4 ThinkingCreature.java

```

package com.kupp.prog3lab;

public interface ThinkingCreature extends Named {
    void think(String about);
}

```

### 3.0.5 Troll.java

```

package com.kupp.prog3lab;

public abstract class Troll extends NamedEntity {
    private TrollTypes type;

    public Troll(String name, TrollTypes type) {
        super(name);
        this.type = type;
    }

    public TrollTypes getType() {
        return type;
    }

    public abstract void interact(String action, Named named);
    public abstract void lookAround();
    public abstract void see(Named named);
    public abstract void ponder();
}

```

### 3.0.6 MoominTroll.java

```

package com.kupp.prog3lab;

public class MoominTroll extends Troll implements ThinkingCreature {
    public MoominTroll(String name, TrollTypes type) {
        super(name, type);
    }

    public void interact(String action, Named named) {
        System.out.printf("%s %s %s.\n", this.getName(), action, named.getName());
    }
}

```

```

public void lookAround() {
    System.out.printf("%s look around.\n", this.getName());
}

public void see(Named named) {
    System.out.printf("%s see %s.\n", this.getName(), named.getName());
}

public void ponder() {
    System.out.printf("%s is thinking about it.\n", this.getName());
}

public void think(String about) {
    System.out.printf("%s think that %s.\n", this.getName(), about);
}

@Override
public boolean equals(Object o) {
    if (!(o instanceof MoominTroll)) {
        return false;
    }
    MoominTroll other = (MoominTroll) o;
    return other.getName().equals(this.getName())
        && other.getType().equals(this.getType());
}

@Override
public String toString() {
    return String.format("MoominTroll with name \"%s\" +
        " & \"%s\" type", this.getName(), this.getType().toString());
}

@Override
public int hashCode() {
    return super.hashCode() + this.getType().getValue();
}
}

```

### 3.0.7 PieTray.java

```

package com.kupp.prog3lab;

public class PieTray extends NamedEntity {
    private int pieCount;
    public PieTray(int pieCount) {
        super("pie tray");
        this.pieCount = pieCount;
    }

    public int getPieCount() {
        return pieCount;
    }

    public void setPieCount(int pieCount) {
        if (pieCount >= 0) {
            this.pieCount = pieCount;
        }
    }
}

```

```

@Override
public boolean equals(Object o) {
    if (!(o instanceof PieTray)) {
        return false;
    }
    PieTray other = (PieTray)o;
    return other.getPieCount() == this.getPieCount();
}

@Override
public String toString() {
    return String.format("PieTray with %d pies", this.getPieCount());
}

@Override
public int hashCode() {
    return this.getPieCount();
}
}

```

### 3.0.8 Box.java

```

package com.kupp.prog3lab;

public class Box extends NamedEntity {
    private BoxTypes type;

    public Box(BoxTypes type) {
        super("box");
        this.type = type;
    }

    public BoxTypes getType() {
        return type;
    }

    @Override
    public String getName() {
        return String.format("%s %s", type.toString(), super.getName());
    }

    @Override
    public boolean equals(Object o) {
        if (!(o instanceof Box)) {
            return false;
        }
        Box other = (Box) o;
        return other.getName().equals(this.getName());
    }

    @Override
    public String toString() {
        return this.getName();
    }

    @Override
    public int hashCode() {
        return this.getName().hashCode();
    }
}

```



```
}
```

### 3.0.9 TrollTypes.java

```
package com.kupp.prog3lab;

public enum TrollTypes {
    EVILDOER("evildoer", 0),
    GOOD("good", 1),
    NERVOUS("nervous", 2);

    private String str;
    private int val;
    TrollTypes(String str, int val) {
        this.str = str;
        this.val = val;
    }

    @Override
    public String toString() {
        return this.str;
    }

    public int getValue() {
        return this.val;
    }
}
```

### 3.0.10 BoxTypes.java

```
package com.kupp.prog3lab;

public enum BoxTypes {
    SQUARE("square", 0),
    ROUND("round", 1),
    TRIANGLE("triangle", 2);

    private String str;
    private int val;

    BoxTypes(String str, int val) {
        this.str = str;
        this.val = val;
    }

    @Override
    public String toString() {
        return this.str;
    }

    public int getValue() {
        return this.val;
    }
}
```

## 4 Результат работы программы

She think that that's funny.  
She think that Sister would be surprised.  
She look around.  
She see pie tray.  
She see round box.  
She is thinking about it.  
She ate 6 pies off pie tray.  
She climbed into round box.

## 5 Вывод

В ходе этой лабораторной работы я поломал голову над реализацией абстрактной объектной модели по тексту, это оказалось непростой задачей для меня, так как пришлось дотраивать модель текста для соответствия работы требованиям. Также я научился вписывать в enum в Java дополнительные методы.