Konyrova Kulyash

1. To complete the first task I learned about inheritance. We had 2 classes, superclass Animal and subclass Rabbit. Subclass inherit from Animal field “name”, but it has it’s own specific field “color”. I override method toString() in Rabbit, so it returns not only name, but color also.

**public** String toString() {

**return** color + " " + **super**.toString();

}

Also I overloaded method Eat(), which can take 1 parameter “amount” or 2 parameters “amount” and “food”.

**public** String Eat(**int** amount) {

**if** (amount > 0)

**return** " pokushal";

**return** " ne pokushal";

}

**public** String Eat(String food, **int** amount) {

**if** ((amount > 0) && (food.equals("morkovka")))

**return** " pokushal morkovku";

**return** " ne pokushal morkovku";

}

1. In second problem i created abstract class «Shape» and it's subclasses Cube, Cylinder and Sphere. There were abstract methods Volume() and areaSurface(), but in each subclass there was a specific formula to find the volune or area of this figure. For example for Cylinder:

**public** **double** surfaceArea() {

// **TODO** Auto-generated method stub

**return** 2\*Math.***PI***\*r\*(h+r);

}

**public** **double** Volume() {

// **TODO** Auto-generated method stub

**return** Math.***PI***\*r\*r\*h;

}

1. In third problem I have 3 classes: person -> employee -> manager. If person has only name, employee has salary, number and year AND name, which was inherited from it’s “father” Person. Class manager is the “son” of Employee, so it has all fields, which Employee has, and specific field bonus. Also, Manager has a Team (Vector), where we can add Employees. In each class I override method toString() and Equals(), so it prints or checks all specific fields of this class. We use keyword “super” which runs method toString() in superclass. (It is in subclass Manager ):

**public** String toString() {

**return** **super**.toString() + " a manager";

}

**public** **boolean** Equals (Manager m) {

**return** **super**.Equals(m) && (m.bonus == **this**.bonus);

}

1. Fourth task was to read a lecture about Polymorphism and write an example. In this task i have one superclass «Person\_poly», which has two subclasses «Student\_poly» and «Employee\_poly». Student is a Person, but Employee is a Person also.
2. In last problem i created abstract class «Piece» with abstract method «isLegalMove()». This superclass has 6 subclasses: King, Queen, Rook, Pown, Knight and Bishop. In each subclass we override method «isLegalMove()» to check. For example, for Rook:

**public** **boolean** isLegalMove(Position newp) {

**if**(newp.x == **this**.x || newp.y == **this**.y)

**return** **true**;

**return** **false**;

}