Filip Vida

Farma

1. Enkapsulácia

```
private double peniaze;
private String meno;
private String priezvisko;
private boolean kupenyTraktor;
private static int pocetDni;
```

```
protected String meno;
protected String pohlavie;
protected boolean dospelost;
protected boolean nakrmene;
protected static int pocet;
```

2. Dedenie

```
public class Zviera {
```

```
public class Sliepka extends Zviera
```

```
public class Nastroj {
```

```
public class Traktor extends Nastroj
```

3. Kompozícia

```
public class Sklad {

    private Sypka sypka ;
```

4. Agregácia

```
public class Farmar {

    private Traktor traktor;
```

5. Prekonávanie

```
public boolean isNakrmene() {
    if(nakrmene)
        System.out.println("Slepka"+this.getMeno()+"je nakrmena a pripravena zniest vajcia");
    return nakrmene;
}
```

```
public boolean isNakrmene() {
    return nakrmene;
}
```

6. Preťažovanie

```
//Vajcia ihned predame
public void pozbierajVajcia(Sliepka sliepky[])
//dame ich do skladu
public void pozbierajVajcia(Sliepka sliepky[], Sklad skladisko)
```

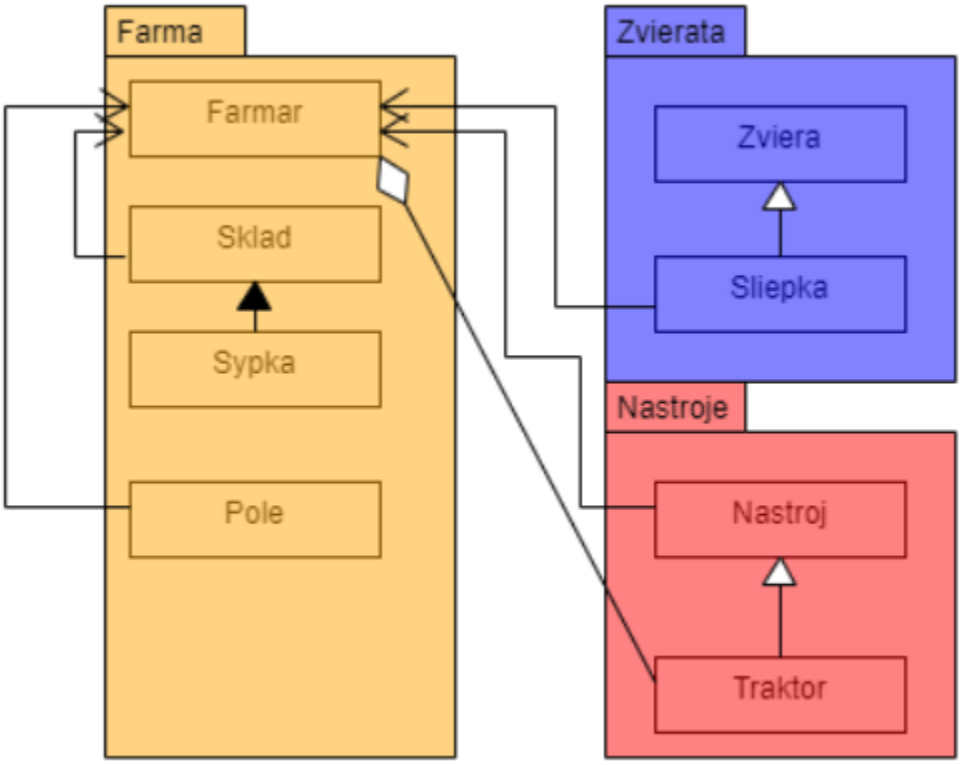
7. Asociácia

```java
public void nakrmSliepky(Sliepka sliepky[], Sklad skladisko)
{
    System.out.println("Na nakrmenie sliepok potrebujeme " + Sliepka.getPocet() + " kukurice");
    if(Sliepka.getPocet()<=skladisko.getKukurica())
    {
    for (int i = 0; i < Sliepka.getPocet(); i++)
    {

        sliepky[i].setNakrmene(true);
        System.out.println("Nakrmil som sliepku "+ sliepky[i].getMeno());
    }
    }
    else
    {
        System.out.println("Nemame dost kukurice v sklade(V sklade je "+skladisko.getKukurica()+")");
    }

}
```

UML mapa



Doplnené nedostatky z 2. Odovzdania

Final metóda (chceme vedieť stale zisti´t stav strojov cez stroj, nechceme ich prepísať)

```
public static final void skontrolujStroj(Nastroj stroj)

public final void getStav()

public final void setStav()
```

3. odovzdanie

1. Polymorfizmus

VytvorTovar() v triede Sliepka

```
public void vytvorTovar()
{
    //lacna vyliahne menej ako normalna atd.
    if(dospelost && nakrmene)
    {
        if(druh == "draha")
        {
            pocetvajec = (int)(Math.random() * ((1 - 4) + 1)) + 1;
            nakrmene=false;
        }
        else if(druh == "normalna")
        {
            pocetvajec = (int)(Math.random() * ((0 - 3) + 1)) + 0;
            nakrmene=false;
        }
        else if(druh == "lacna")
        {
            pocetvajec = (int)(Math.random() * ((0 - 2) + 1)) + 0;
            nakrmene=false;
        }
    }
}
```

VytvorTovar() v triede Krava

```
public void vytvorTovar()
{
    if(nakrmene && dniOdDojenia>=2 && mlieko == 0)
    {
        if(druh == "draha")
        {
            mlieko = 4;
            nakrmene=false;
            pripravenaNaDojenie = true;
        }
        else if(druh == "normalna")
        {
            mlieko = 3;
            nakrmene=false;
            pripravenaNaDojenie = true;
        }
        else if(druh == "lacna")
        {
            mlieko = 2;
            nakrmene=false;
            pripravenaNaDojenie = true;
        }
        System.out.println("Krava " + this.meno + " je pripravena na dojenie");
    }
}
```

2. Downcasting

```
public void nakrm(Zviera zver[], Sklad skladisko)
{
    if(zver[0] instanceof Sliepka)
    {
        Sliepka sliepocka = null;
        System.out.println("Na nakrmenie sliepok potrebujeme " + Sliepka.getPocet() + " kukurice");
        if(Sliepka.getPocet()<=skladisko.getKukurica())
        {
            for (int i = 0; i < Sliepka.getPocet(); i++)
            {
                sliepocka = (Sliepka) zver[i];
                sliepocka.setNakrmene(true);
            }
        }
        else
        {
            System.out.println("Nemame dost kukurice v sklade(V sklade je "+skladisko.getKukurica()+")");
        }
    }
    else if(zver[0] instanceof Krava)
    {
        Krava kravicka = null;
        System.out.println("Na nakrmenie krav potrebujeme " + Krava.getPocet() + " psenice");
        //System.out.println(Krava.getPocet());
        if(Krava.getPocet()<=skladisko.getPsenica())
        {
            for (int i = 0; i < Krava.getPocet(); i++)
            {
                kravicka = (Krava) zver[i];
                kravicka.setNakrmene(true);
            }
        }
        else
        {
            System.out.println("Nemame dost psenice v sklade(V sklade je "+skladisko.getPsenica()+")");
        }
    }

}
```

3. Upcasting

```
public static void skontrolujStroj(Nastroj stroj)
{
    System.out.println("Nazov tohto nastroju je " + stroj.getNazov());
    if(stroj instanceof Traktor )
    {
        System.out.println("Je to traktor");
    }
    else if(stroj instanceof Traktor)
    {
        System.out.println("Je to kombajn");
    }
    System.out.println("Nazov tohto nastroju je " + stroj.getNazov());
    stroj.getStav();

}
```

```
Nastroj.skontrolujStroj(filip.getTraktor());
```

4. Abstraktná classa s 2 abstraktnými metódami

```
public abstract class Zviera {
    protected String meno;
    protected String pohlavie;
    protected boolean dospelost;
    protected boolean nakrmene;
    protected String druh; //lacna, normalna, draha

    public String getMeno() {
        return meno;
    }

    public String getPohlavie() {
        return pohlavie;
    }


    public boolean isDospelost() {
        return dospelost;
    }

    public boolean isNakrmene() {
        return nakrmene;
    }

    public abstract void setNakrmene(boolean nakrmene);

    public abstract void vytvorTovar();


}
```

5. Interface

```
public interface Zem {
    public void zasad(String druh);
    public void dozrej();
    public void zatva();
}
```

UMl