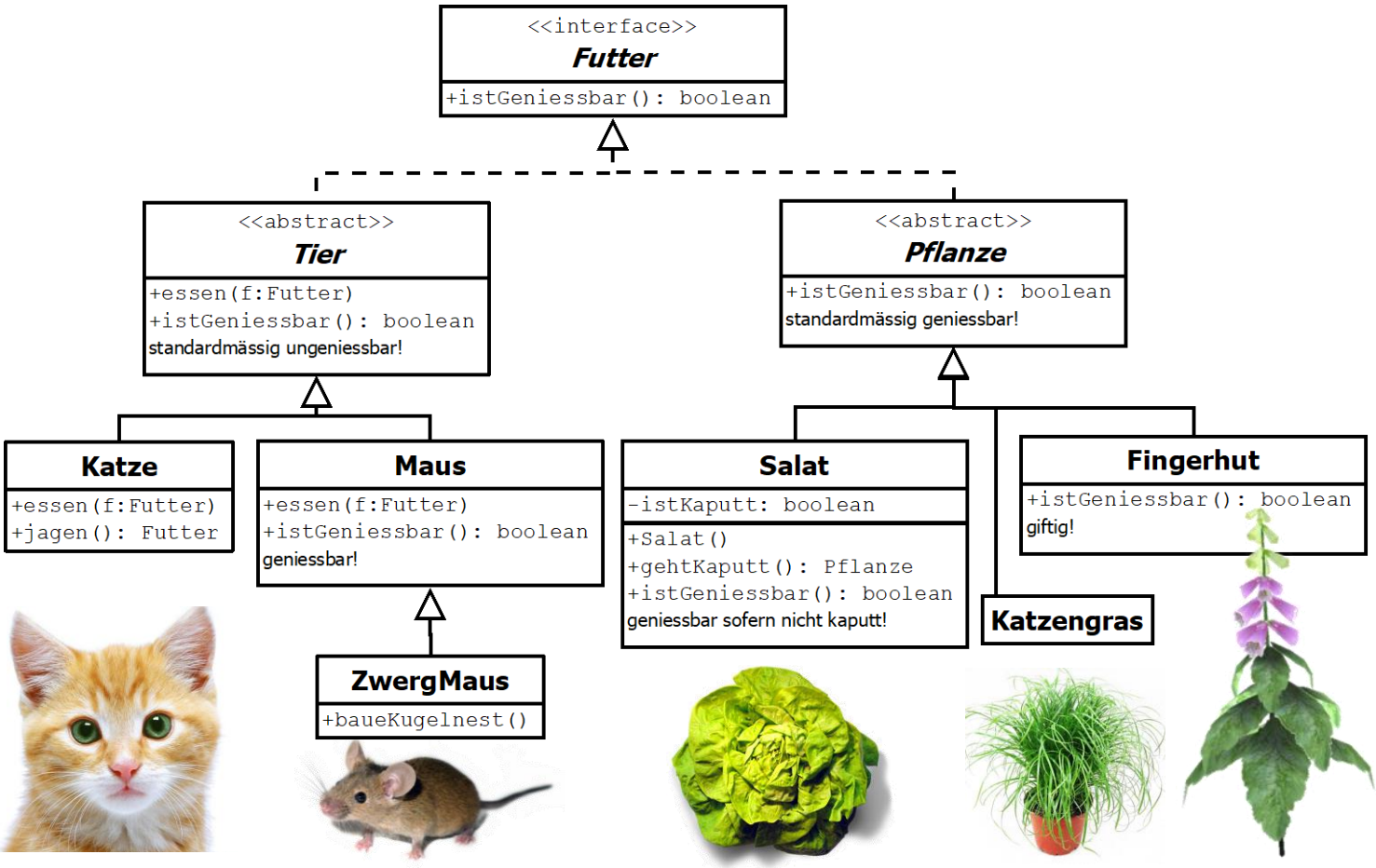
1.
Laden sie die „**Tierpark**"-Klassen (Paket **park**) in ihren Workspace und untersuchen sie das dazu passende UML-Klassendiagramm:



Vervollständigen sie alle mit TODO-gekennzeichnten Stellen der Klassen **`Katze`**, **`Maus`**, **`Zwergmaus`**, **`Salat`**, **`Fingerhut`** und die **`main`**-Methode der Klasse **`Tierpark`**, anhand des UML-Klassendiagramms und folgender **Anforderungen:**

- **`Salat`** kann verderben, d.h. nach Aufruf der Methode **`gehtKaputt`** ist er nicht mehr geniessbar.
- **`Fingerhut`** ist nicht geniessbar, da giftig.
- **`Katzen`** essen nur **`Mäuse`** oder **`Katzengras`**
- Die **`Katze`** "jagt" solange, bis sie passendes **`Futter`** findet, d.h entweder eine **`Maus`** fängt, oder auf **`Katzengras`** trifft.
- **`Mäuse`** haben Fressfeinde und sind daher "***geniessbar***"
- **`Mäuse`** essen nur **`Pflanzen`**

Zusatzaufgabe - Reflection

2.
Legen sie ein Paket namens **`reflect`** an und schreiben sie ein Programm, dass die vorgegebene Klasse „**`reflect.UnbekannteKlasse`**" zur Laufzeit einbindet, diese analysiert und alle Attribute, Konstruktoren, Methoden textuell ausgibt.

Hinweis: Zur Lösung lesen sie in der JDK-API-Hilfe bei den Klassen **`Class`**, **`Field`**, **`Constructor`** und **`Method`** nach.

```
public class TierPark {

      // statische Hilfsmethode zur Futtererzeugung
      public static Futter getFutter(){
        // Es wird zufälliges Objekt einer "Futter"-Implementierung geliefert
        int i = (int)(Math.random()*8);
        switch(i){
            case 0: return new Fingerhut();
            case 1: return new Maus();
            case 2: return new ZwergMaus();
            case 3: return new Salat().gehtKaputt(); // Salat schon verdorben
            case 4: return new Katzengras();
            case 5: return new Katze();
            default: return new Salat();
        }
      }


      public static void main(String[] args) {
        // Tier-Array anlegen
        Tier[] zoo = { new Katze(), new Maus(), new ZwergMaus(), new Maus(),
                       New Katze(), new Maus() };
// TODO: Über Tier-Array iterieren und folgendes tun:
// - Klassenname jedes Tieres ausgeben
// - Katzen-Objekte "jagen" erst selbst und essen dann das gejagte Futter
// - ZwergMaus-Objekte bauen vor dem Essen erst ihr Kugelnest
// - alle anderen Tiere essen das Futter vom Tierpark
//   (siehe Methode getFutter())
```

```
}
```

```
//-------------------------------------------------------------------
interface Futter{
      boolean istGeniessbar();
}

//-TODO Kopf ---------------------------------------------------------
         class Pflanze                                    {
      public boolean istGeniessbar(){
            // Pflanzen sind per default geniessbar
            return true;
      }
}

//-TODO Kopf---------------------------------------------------------
class Salat                                    {
      private boolean istKaputt;
// TODO: Es gibt auch Salat der kaputt gegangen ist…
```

```
      public Pflanze gehtKaputt(){
// TODO:


            return this;
      }
}

//-TODO Kopf ---------------------------------------------------------
class Fingerhut                      {
      // Fingerhut ist giftig!
      public boolean istGeniessbar(){
            return false;
      }
}

//-TODO Kopf---------------------------------------------------------
class Katzengras                       {
}

//-TODO Kopf---------------------------------------------------------
       class Tier                                       {
      public void essen(Futter f){
        // nur geniessbares Futter wird gegessen
        if(f.istGeniessbar()){

          System.out.println("Tier.essen:+ this.getClass().getSimpleName()+
                             " isst "+ f.getClass().getSimpleName());
        } else {
          System.out.println("Tier.essen: Futter " +
                            f.getClass().getSimpleName()+ " ungeniessbar!");
        }
      }
      public boolean istGeniessbar(){
            return false; // Tiere sind per default nicht geniessbar
      }
}
```

```
//-TODO Kopf-------------------------------------------------------
class Katze                     {
      @Override
      public void essen(Futter f) {

// TODO: Katzen essen nur Mäuse oder Katzengras
        if (                                                        )

// TODO: Aufruf essen der Basisklasse


        else
            System.out.println("Katze.essen kein passendes Futter" +
                               f.getClass().getSimpleName());
      }

      public Futter jagen(){
        Futter f=null;
// TODO:  Die Katze jagt solange, bis sie eine Maus fängt, oder auf
// Katzengras trifft. Zufällige Futter-Objekte koennen mit
// Tierpark.getFutter() abgefragt werden.









        return f;
      }
}

//-TODO Kopf-------------------------------------------------------
class Maus                          {
// TODO: Mäuse haben Fressfeinde und sind daher "geniessbar"









// TODO: Mäuse essen nur Pflanzen









}
//-TODO Kopf-------------------------------------------------------
class ZwergMaus                             {
      public void baueKugelnest(){
        System.out.println(getClass().getSimpleName() + " Kugelnest gebaut");
      }
}
```