You are given the task of creating a computer model of Cluj's future zoo, creatively named ***Zoowsome***.

There are quite a few steps in modelling this zoo, however, as an overview, the zoo will have *animals*, *employees*, *tourists* and *attractions*. Abstracting all these things in your code will be a quite a task, but everything will be done step-by-step.

First of all, you will need 4 packages namely:

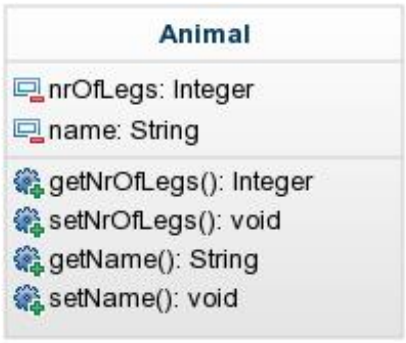
- javasmmr.zoowsome.models
- javasmmr.zoowsome.views
- javasmmr.zoowsome.controllers
- javasmmr.zoowsome.services

In this first week we will mostly be dealing with the *models* package. (For more information on *package naming* read up here, here and of course, feel free to use Google)

The *models* package will have two *sub-packages*:

- animals *(javasmmr.zoowsome.models.animals)*
- employees *(javasmmr.zoowsome.models.employees)*

You will need to create an abstract *Animal* class. This abstract class will contain attributes (and their getters/setters) which all animals will have. For example, *nrOfLegs*, *isMammal*, *name*. 2attributes should do, for now. The UML diagram for this class looks like this:



Keep in mind, it is an abstract class – you cannot create any instance of it, however it can contain non-abstract methods and can be *extended*. If you do not yet understand what an abstract class is, make sure to read-up on it before going further.

Yes, all animals may have a name and a predefined number of legs, but there are at least 6 (based on a basic classification) types of animals: *mammals*, *reptiles*, *birds*, *aquatic* (incl. amphibians) and *insects*. Each one of these 5 has specific attributes:

- Mammal:
  - Normal body temperature (normalBodyTemp - float)
  - % of body covered in hair (percBodyHair - float)

- Reptile:
  - Does it lay eggs or not (laysEggs - Boolean)
- Bird:
  - Does it migrate or not? (migrates - Boolean)
  - Average flight altitude (avgFlightAltitude - Integer)
- Aquatic:
  - Average swim depth (avgSwimDepth – Integer)
  - Saltwater or freshwater (waterType – Enum) – construct special enum for this
- Insect:
  - Can it fly? (canFly – Boolean)
  - Is it dangerous? (isDangerous – Boolean)

For each one of these animal types, create a new *abstract* java class with the corresponding attributes, following the model of the *Animal* class. **Each of these animal sub-classes must extend the *Animal* class!**
Now bear with me: create at least 3 subclasses of each of the above subclasses. For example:

- Mammal
  - Tiger
  - Monkey
  - Cow
- Insect
  - Butterfly
  - Spider
  - Cockroach
- Etc.

So, at least 15 new classes should be crated, each extending its proper parent class.

public class Tiger extends Mammal { … }
public class Butterfly extends Insect { … }
…

Make sure to create a constructor for each of these classes and set all the attributes inside of it. Also provide a *no-arguments* constructor for each classes and set the attributes to some predefined values. (ex: for Cow, *nrOfLegs* should be 4)

Now we need to create some sort of way to populate our zoo. This might at first look very cumbersome at first, but please, bear with me and you'll see beauty in design at the end.

In the *javasmmr.zoowsome.services* package, create a sub-package called *factories* (javasmmr.zoowsome.services.factories).
In this package, create a class called *AnimalFactory*. This class will have just one method:

```
public SpeciesFactory getSpeciesFactory(String type) {
      if (Constants.Species.Mammals.equals(type)) {
            return new MammalFactory();
      } else if (Constants.Species.Reptiles.equals(type)) {
            return new ReptileFactory();
      } else if (Constants.Species.Birds.equals(type)) {
            return new BirdFactory();
      } else if (Constants.Species.Insects.equals(type)) {
            return new InsectFactory();
      } else if (Constants.Species.Aquatics.equals(type)) {
            return new AquaticFactory();
```

```
                } else
                        throw new Exception("Invalid species exception");
        }
```

You may notice that there is something like *Constants.Speciess.Mammals* in the code snippet from above. You can find this whole class in the appendix on the last page. Make sure to modify based on your needs.

The next class you will have to create in this package is *SpeciesFactory*. This will be an **abstract** class and the whole code for this class is:

```
public abstract class SpeciesFactory {
        public abstract Animal getAnimal(String type);
}
```

You will have to also create 5 different classes for each of the 5 animal types (Mammal, Bird etc.) They should follow this naming convention: *MammalFactory*, *ReptileFactory* etc.
They should all extend the species factory and they will look like this:

```
public class MammalFactory extends SpeciesFactory {

        @Override
        public Animal getAnimal(String type) {
                if (Constants.Animals.Mammals.Dog.equals(type)) {
                        return new Dog(); // leave empty constructor, for now!
                } else if (Constants.Animals.Mammals.Cat.equals(type)) {
                        return new Cat();
                } else {
                        throw new Exception("Invalid animal exception!");
                }
        }

}
```

Based on all the actual (non-abstract) animal classes you created, you will have to fill the above *getAnimal* method accordingly, for each of the 5 factories.

We're almost done for this week, now let's check out our design and classes.
In *javasmmr.zoowsome.controllers* create a class called *MainController* in which you'll have to setup your *main* method. In there, simply try out this snippet of code:

```
        public static void main(String[] args) {
                AnimalFactory abstractFactory = new AnimalFactory();

                SpeciesFactory speciesFactory1 =
abstractFactory.getSpeciesFactory(Constants.Species.Mammals);
                Animal a1 = speciesFactory1.getAnimal(Constants.Animals.Mammals.Cat);
                System.out.printf("We have an animal with %d legs!\n",
a1.getNrOfLegs());

        }
```

Feel free to play around with creating as many animals as you want, from all different species and print out anything you want from them. You can also try to play around with casting from one class to another. See which way does it work? From subclass to superclass? Or the other way around? Or both?

Twist1:
Randomize the creation of animals in the main method. Create some sort of counter which can be set to 50 for example and at the end of everything, you will have created 50 random animals from each species and types.

Twist2:
In the factories, find some ingenious way of randomizing the attributes passed in each constructor when creating an animal. Of course, you'll have to have a specific randomization method for each attribute and animal type, but feel free to improvise as much as possible here. Use your imagination.

```java
/**
 * @author Alex Holder class for string
 */
public final class Constants {

	public static final class Species {
		public static final String Mammals = "Mammals";
		public static final String Reptiles = "Reptiles";
		public static final String Birds = "Birds";
		public static final String Aquatics = "Aquatics";
		public static final String Insects = "Insects";
	}

	public static final class Animals {

		public static final class Mammals {
			public static final String Cat = "CAT";
			public static final String Dog = "DOG";
			public static final String Wolf = "WOLF";
		}

		public static final class Reptiles {
			public static final String Lizzard = "LIZZARD";
			public static final String Snake = "SNAKE";
			// ...
		}

		public static final class Birds {
			// TODO: complete this yourself
		}

		public static final class Aquatics {
			// TODO: complete this yourself
		}

		public static final class Insects {
			// TODO: complete this yourself
		}
	}

}
```