SYSC 2004 C and D: Object-Oriented Software Development **Lab No 9:** [**1** mark]

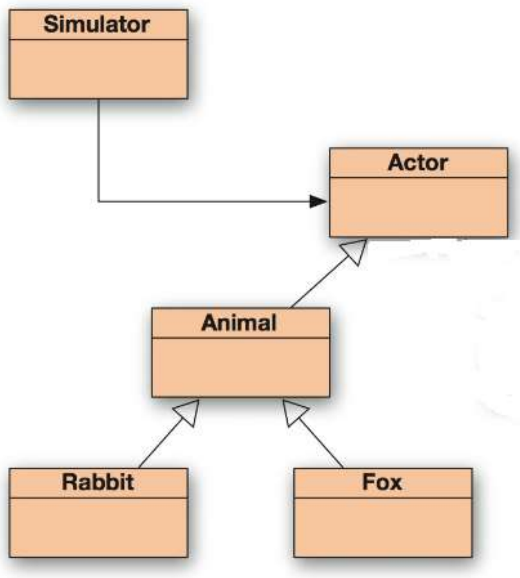
The main objective of this lab No 9 is to learn about *abstract classes and interfaces* and how to implement them effectively in Java. To this end we will be reusing the *fox-and-rabbits-v2* project from Chapter 12. Source code for this project is available in cuLearn in the general block (Chapter 12). Open the *fox-and-rabbits-v2* project.

1. Introduce the **Actor** class into your simulation. Rewrite the **simulateOneStep** method in **Simulator** to use **Actor** instead of **Animal**. You can do this even if you have not introduced any new participant types. Does the **Simulator** class compile? Or is there something else that is needed in the **Actor** class? The **Actor** and **Animal** classes are *abstract* classes, while **Rabbit** and **Fox** are *concrete* classes. The **Actor** class would include the common part of all actors. One thing all possible actors have in common is that they perform some kind of action. You will also need to know whether an actor is still active or not. So, the only definitions in class **Actor** are those of abstract **act** and abstract **isActive** methods:

```
// all comments omitted
public abstract class Actor
{
    abstract public void act(List<Actor> newActors);
    abstract public boolean isActive();
}
```

You need to rewrite the actor loop in the **Simulator**, using **Actor** instead of class **Animal**. (Either the **isAlive** method could be renamed to **isActive**, or a separate **isActive** method in **Animal** could simply call the existing **isAlive** method).

Figure 1
Simulation structure with **Actor**



2. Redefine as *interface* the abstract class **Actor** in your project. Does the simulation still compile? Does it run? Make any changes necessary to make runnable. Classes (such as Animal) that *extend* **Actor** will now have to *implemen*t it instead.

You need to document your classes and their methods using the Javadoc tool integrated with BlueJ.

To test your work, you need to create **a Simulator** object, using the constructor without parameters in the object bench. Call the **simulateOneStep** method just once. Call the **simulate** method with a parameter to run the simulation continuously for a significant number of steps, such as 50 or 100. You can use the **runLongSimulation** method. Test that all methods work as expected.

Important Notes and Submission Guidelines:

You start first by identifying the classes (in the inheritance hierarchy), reading the source code of *fox-and-rabbits-v2* project and identifying objects and methods involved in every functionality.

You have to do and pass a demo using BleueJ with your teaching assistant (TA) in the lab room or during her/his office hours. Your TA will lead you and conduct and run the tests of your programs and evaluate your work. You are also allowed to do your demo using your own laptop.