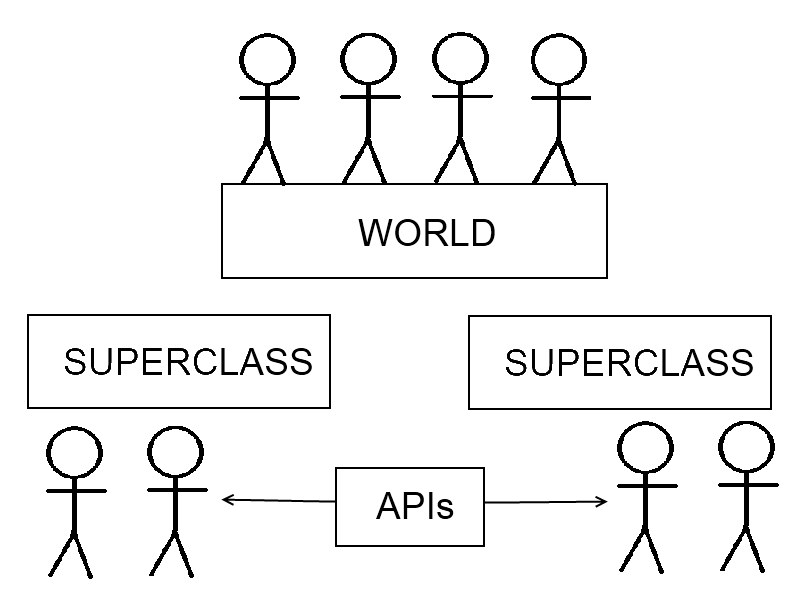
OOP Simulation – Group Project

In groups of 4 students, you will create a **Greenfoot simulation**. Your project will include the following:

* **A subclass of World** – your own World, which will act as the **controller** for your simulation. This class will be responsible for spawning and managing Actors, as well as coordinating communication between Actors.
* **At least two abstract parent classes** – there must be at least two major categories of objects which will interact in your game (For example, in the Vehicles example, there were Vehicles and Pedestrians). Each parent class will have at least two (2) child classes (I.e. Bus, Car and Ambulance were the three children of Vehicle). The parent classes should NOT be similar to one another.
* **Class diagrams** for the classes in your program
* Well documented **API**s for each Actor as well as the World in your simulation
* **A 5-minute presentation** to demonstrate and explain your simulation to the class

# Instructions

1. Read the entire assignment carefully.
2. As a group, come up with a topic for your simulation. Ideas may be found in the world around you, in business, in nature, etc.
3. Confirm **topic** with your teacher. If (topic.isRejected()){return to step 1;}
4. Complete a **proposal**, (as explained below). Divide into subgroups
5. Create your **World** and change the resolution to approximately 960x640 (this is about the largest size that will comfortably fit on the LCD projector, for your presentation)
6. Design a **class structure** for your program – this can be changed as you go, but it is very important to plan ahead. Next, create an **API** for each Class in your project, paying special attention to the child classes.
7. As a group, code the **basic functionality** needed for your **World**, based on the class diagram. (Note – there will not be much to do at this point, you will add methods, variables and code to your World as you go.)
8. **Divide into two subgroups** of two students each. Each sub-group will be responsible for designing and coding one abstract parent and all of its children. In order to ensure that both sub-groups’ classes work well together, continue to develop your API as you go, sharing it regularly with your group. (For example, in our Vehicles simulation, if a vehicle hits a pedestrian, it calls the knockMeOver() method. If you share your API, you will know what methods need to be called, their name and their parameters). (If one superclass has a lot more content than the other, then help each other out as necessary).
9. Once all classes are complete, work together to make necessary modifications to the World to ensure your objects interact properly, and your simulation works as you envisioned. If time remains, focus on improving the quality, depth and interest of your simulation.
10. Create **good copy** of class diagrams and APIs, to be submitted. See submission instructions below for a checklist.
11. Plan a **5-minute presentation** that will demonstrate the simulation and its features to your class.

**Proposal:**

To begin, complete the back page and have your teacher initial it, including:

* Your topic, which has already been approved by your teacher
* Your super-classes (part of the topic, which was approved), and a few ideas for child classes
* List of full names of students in the group, divided into two-student subgroups, organized by which parent class you are working on.

**Final Submission:**

* Class Diagram
* APIs for all classes, and your World
* Code (Submission instructions will be given)

**Presentation:**

* Your group will perform a 5 minute presentation about your simulation
* Topics to be discussed include:
  + Explain what your simulation will do before you show it to the class
  + Allow the class to watch your simulation for about 1 minute (Your simulation should be interesting to watch for a minute!)
  + Discuss the how elements of object oriented programming, APIs and class diagrams improved your simulation, both **product** and **process**.

# Notes:

* You have approximately 6 periods to complete this assignment. Ensure that the group sets deadlines in order to stay on track. There will be **deductions** for any group not ready to submit paperwork, code and perform their presentation on the **due date**.
* The World is the controller – all communication between methods should take place either in the World, or directly by collision. Keep in mind, that if you need to keep track of overall numbers (I.e. how many pedestrians have been run over?) you should insist that all communication takes place through the World
* Most Actors in the World should be **owned** by references in your World class (So, should avoid adding important objects straight into the World without initializing them first). You can use arrays to hold multiple, related objects. Remember, arrays can hold objects of different subclasses as long as they share a superclass. See the following code example for GOOD and BAD practices:

**// Owning a reference - GOOD**

**Elephant e = new Elephant (2, “Sam”);**

**addObject (e, x, y);**

**// Adding an unreferenced object – BAD!**

**addObject (new Elephant(2, “Sam”), x, y);**

* Your simulation should be interesting to watch (for at least a minute). Try to create interest by adding a varied set of Actors and interactions.
* Use graphics and sound appropriately to bring your simulation to life.

**Topic:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |  |  |
| --- | --- | --- | --- |
| **Abstract Parent Class #1** | **Class:** | **Names:** | |
|  | **Details:** | | |
| **Abstract Parent Class #2** | **Class:** | | **Names:** |
|  | **Details:** | | |

**Marking Scheme:** Teacher Initials: \_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| **Thinking/Inquiry** | Planning, Class Diagrams, API, Documentation  Good programming practices | / 10  / 10 |
| **Communication** | Presentation  Audio and Visual Elements, Creativity | / 10  / 10 |
| **Application** | Good use of OOP concepts, including proper use of inheritance and object oriented design | / 10 |

**Teacher Comments: / 50**