

**Victoria University of Technology**  
**School of Computer Science and Mathematics**

**SCM2311 Object Oriented Programming 1**

**Assignment #2 (20%)**

**Due: 6<sup>th</sup> June 2005**

**Project Overview**

Go to the subject web page <http://melba.vu.edu.au/~scm2311> and download assignment2.zip that contains source code to simulate three fish swimming in an aquarium. For now, our aquarium representation is quite simple. Compile all classes and run the driver class *ThreeSwimmingFish.java* to get a feel of simulation of fish movement in an aquarium.

Study the classes and interface carefully.

*Aquarium.java* – represents an aquarium

*Fish.java* – represents a fish

*Moving.java* – represents a moving fish

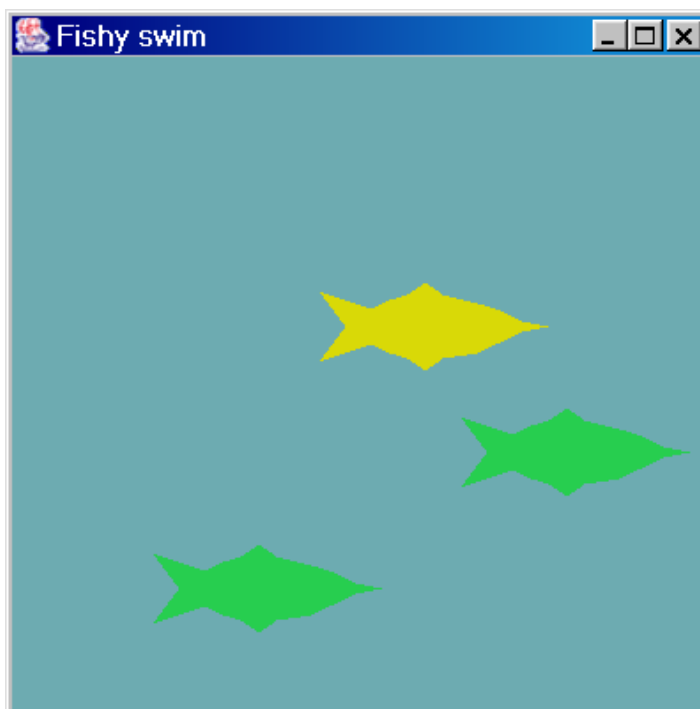
*Step.java* - generates changes in x and y from specified intervals

*ThreeSwimmingFish.java* – a driver program to simulate three swimming fish

*Sleep.java* – supports pausing of animation to allow the movement to be perceived

*Drawable.java* – an interface to return the x and y coordinates of the object and to draw the object

Below is a snapshot of *ThreeSwimmingFish.java* display.



## Task

The first thing you must do is to find a partner of the same lab group. Make sure you choose someone who will share the work equally with you. Together, you are to design and implement a Java application (Part I) to create a more realistic aquarium simulator.

## Part I (80 marks)

The application shall provide:

- A thread to represent each swimming fish.
- Define a class called **BetterFish** where the fish has the capability to change course when it reaches the limit of its territory. In support of this development, you will need to design and implement classes **Territory** and **BoundaryBehavior**.
- Design and implement a class **UnderwaterPlant** for representing plants. Provide a method **makePlant()** that creates a basic underwater plant.
- Design and implement a class **Aerator** for representing an aquarium air bubbler. Provide a method **makeAerator()** that creates a basic aerator. An aerator release bubbles in a regular manner that slowly rise to the surface.
- Lastly, develop a program **ThreeSwimmingBetterFish.java** that demonstrates your fish in action.

## Some ideas to help break down the problem

Remember to look for classes that share responsibilities, capabilities, or interfaces and make an interface or superclass for them.

1. Class **Territory** supports objects that maintain and give access to four values:  
**private int xMin**  
**private int xMax**  
**private int yMin**  
**private int yMax**

These instance variables should also be used in implementing the following behaviours:

//indicates whether location (x,y) lies inside or on the edge of the territory  
**public boolean inside (int x, int y)**

//indicate whether location(x,y) lies neither inside nor on the edge of the territory  
**public boolean outside (int x, int y)**

//indicate whether location(x,y) lies on the edge of the territory  
**public boolean atLimit(int x, int y)**

2. Class **BoundaryBehavior** should provide four public class constants to correspond to common behaviours with regards to an object's continued motion when it reaches a vertical or horizontal boundaries:
  - i. **BoundaryBehavior.Freeze:** When a vertical or horizontal boundary is reached by the object, its motion pauses.

- ii. **BoundaryBehavior.Reverse:** When a vertical or horizontal boundary is reached by the object, motion continues in the opposite direction from the current position.
  - iii. **BoundaryBehavior.Clockwise:** When a vertical or horizontal boundary is reached by the object, motion continues from the current position in the direction that is clockwise to the reached boundary.
  - iv. **BoundaryBehavior.CounterClockwise:** When a vertical or horizontal boundary is reached by the object, motion continues from the current position in the direction that is counter-clockwise to the reached boundary.
3. The class **BetterFish** should at a minimum provide the following constructors:
- ```
//create a fish with default shape, position and movement. The territory of the fish
//should be t and its boundary behaviour should be b.
public BetterFish (Territory t, BoundaryBehavior b)
```
- ```
//Creates a fish with color c, shape p at coordinate(x,y), movement s, territory t,
//and boundary behavior b.
public BetterFish(Color c, Polygon p, int x, int y, Step s, Territory t,
BoundaryBehavior b)
```
- ```
//override the swim( ) method to respond appropriately when a stroke would bring a
// fish outside of its territory
public void swim( )
```

## **Part II –Enhancements (20 marks)**

You are encouraged to enhance the basic product. A maximum of 20 marks will be allocated to this.

You may choose to extend Part I to include features to produce:

For example,

- Better graphics for different species of fish.
- Modify the program so that the fish is initiated by a **mousePressed** event. Have the fish begin at the point where the mouse was pressed.
- Design and implement a class **Sand** for representing the gravel surface often found in aquariums. You may want to provide a method **makeSand( )** to do this. Incorporate the presence of sand in your testing.
- Redesign the class **Aquarium** so that it supports different-sized aquariums.
- Anything interesting that you would like to add. Be creative! Have fun!

## **Restrictions on Design and Implementation**

1. This assignment is to be written as a **java application** using JDK1.5 and **Swing** classes wherever possible.
2. Follow all the requirements of the assignment by using multiple classes in separate files using object-oriented principles such as decomposing code into a number of classes and encapsulating repeated code in common classes.
3. Separate the model and view.

## **Deliverables**

You need to demo your program to the tutor during the lab session. In addition, you should include hard copies of the following:

1. A high level class diagram, showing the relationships between classes from the *design* viewpoint.
2. Individual class diagrams for each class in the system.
3. A description of how your program works from the *operation* viewpoint. How can the user perform each of the operations required?
4. A printout of the full system documentation of class definitions using javadoc.
5. A printout of your source code.
6. A virus free diskette containing all the source code (java files) with appropriate comments at the start of each file and at the start of each method, compiled code (class files), and javadocs in a folder. Make sure you write down your name, your partner's name and student Ids, subject code and date written on your hard copy as well as on your diskette.

Marks will be awarded on the basis of your design, the program's correctness, clarity, use of comments, completeness and efficiency. Be disciplined and persistent when you test and debug.

## **Academic Integrity**

You may discuss the assignment with other classmates at a high level. However, you may not share your code or your design with others. All work must be the original work of the submitting group. Any unauthorized collaboration or copying will result in no credit for the assignment.

## **Late Submissions**

If you do not submit your assignment on time, a penalty of 20% of the total awarded marks per academic day will be applied. Assignment will not be accepted 2 academic days after the due date.