Extended Java Lab Assignment 3:

This assignment covers generics and multithreading. It is out of 30 marks, 20 for question 1 and 10 marks for question 2. You must complete your answer using the IntelliJ template provided. In both questions the style and quality of your code will be taken into consideration when allocating marks.

Part 1: MiniList collection [20 marks]

In this question you are going to produce a simplified collection type called `MyMiniList`. Use the template on learn, gold to get started. MyMiniList will work in a similar way to ArrayList. Internally, it will use an array, you will have to reorganise and grow the array as the MiniList is updated through use of its methods.

Inside the template is an interface called MiniList. MyMiniList must implement this interface and be implemented with a single generic type.

Also included in the template is a simple set of tests for you to verify your implementation against. You will receive the following marks for correctly implementing each of the interfaces methods and a constructor.

Your class will need two instance variables, 1 for the current size of the list (an int) and an array of generic type T to store the elements of the list call this objectStore.

public MyMiniList() [2 marks]

Initialise the array of elements, this is a little tricky as Java Generics doesn't support arrays. There are a couple of approaches you can use. This is the easiest and is moderately type safe, however, IntelliJ will give you some warnings:

```
objectStore = (T[]) new Object[10];
```

Initially, set the size of this array to be 10. Also set your size variable to be 0.

public void add(T element) [3 marks]

Elements are added to the end of the objectStore array if the array has space. If the array doesn't have space you will need to create a new array twice the size now required, copy in the existing elements and replace the existing objectStore with this new array. System.arrayCopy is useful to achieve this but there are other methods you could use. You can then add in the new element and update the size variable.

public T get(int index) [2 marks]

Check that the index given is with the required bounds. You can use the built in method Objects.checkIndex to do this (no imports required). If the index is valid return the item at the index.

public int getIndex(T element) [2 marks]

Search through the list for the first occurrence of the element given as an argument. If found return its index. Otherwise return -1.

public void set(int index, T element); [2 marks]

Check if the index is valid. If so set the value at the index to be the element given in the second parameter.

public int size(); [1 mark]

Return the size of the list. (Make sure you return the size of list not the size of the underlying array)

public T remove(int index); [4 marks]

Check if a valid index has been given. If so remove the element and return it. You will need to move down the end of the array after the removal point so there is no gap and adjust your size property accordingly.

public boolean remove(T element); [4 marks]

Search the list for the first instance of the element and remove it. Again you will need to shift the array down to fill the gap left. TIP: you might be able to use the previous method to help you.

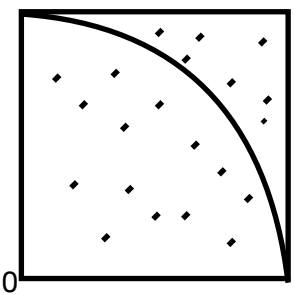
public void clear(); [1 mark]

Empty the list and reset the size to 0.

Part 2 Multithreaded pi estimation [10 marks]:

In the class PiEstimation is a simple algorithm that makes an estimate of Pi. The algorithm works by throwing theoretical darts at a dart board like the one in the drawing below, darts that hit the board fall within the quarter circle darts that miss are outside it. The dartboard has a height and width of 1, this makes our calculations easier. We throw a dart by generating a random x and y coordinate for the dart between 0 and 1. A dart is said to hit the board if the distance from the origin is less than 1, we can calculate this using pythagorus ($x^2+y^2 = dist^2$) If we take the proportion of darts that hit the board (divide the number of hits by the total number of darts thrown) and multiply by 4 we get a surpassingly accurate estimate for Pi.

This estimate only works because we have thrown a lot of darts. Rewrite the application so that it can be run across multiple threads. You answer should include the following:



- A TotalWithin class that keeps track of the number of darts that are within the dart board
- A class that extends Thread which throws darts, works out their distance from the origin and increases the value held within TotalWithin if needed.
- Test your programme with 4 threads running and get an estimate of pi based on 1 billion darts.