Name_____ Student ID_____

1. (a) What OO Design Principle is evident in the application of Iterator Pattern on different types of collections? [2 pts]

(b). One major concern about implementing an interface to traverse through a collection is performance. How is this or how may this be effectively addressed in the implementation of Iterator pattern. [2 pts]

(c). One may simply provide methods on the collection to navigate through it. Why should we provide (and use) an Iterator instead? [2 pts]

(d). What creational patter may be used for different types of collections that inherit say from a Collection interface or abstract class to return different types of Iterators? [2 pts]

2. What is the difference between object scope adapter and class scope adapter? What are the benefits and disadvantages that one offers over the other? [5 pts]

3. Mention a behavioral design pattern that promotes composition instead of inheritance. Explain. [3 pts]

4. In Java, a derived class overriding method is not allowed to throw any new exception that the corresponding base class method does not throw. Which design principle is evident here? Explain what will be the consequence if this principle were not being enforced. [5 pts]

5. (a). Why is it not a good idea to use run time type identification? What principle(s) is/are violated if we use it indiscriminately? [3 pts]

(b). Is there a case where using RTTI would not be an issue? Give an example code in a language of your choice of such a usage. [3 pts]

6. (a). What does the Acyclic Dependency Principle state? [1 pts]

   (b). Why is this principle important? [2 pts]

   (c). Provide suggestions on how one may break cyclic dependencies. [3 pts]

7. Which recommendations in Extreme Programming promote code review and constant feedback? [2 pts]

8. Given the following code [15 pts]:

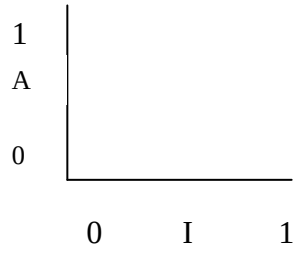
```
// Assume Engine and TurboEngine are both part of a Engines package (namespace in C++).
package Engines;
public class Engine {…}

package Engines;
public class TurboEngine extends Engine {…}

import Engines.*;
public class Car
{
        private Engine theEngine;
        …
        public Car(Engine anEngine) { theEngine = anEngine; }
        public Car(Car other)
        {
                if (other.theEngine instanceof TurboEngine)
                        theEngine = new TurboEngine((TurboEngine)(other.theEngine));
                else
                        theEngine = new Engine(other.theEngine);
        }
}
```

(a). Draw one UML diagram showing the relationship between the above classes

(b). Derive (show how you get these values) the value of I (the Instability value), the value of A (the Abstraction value) and the value of D' (normalized distance from main sequence = |A + I – 1|) for each package. Plot these values for the two packages on the graph of A vs. I (I on x-axis). What is the average D' value of the code above.

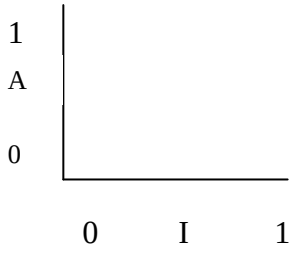


(c). What principle is being violated in the above code? Explain.

(d) How would you solve the above problem (Explain – no diagrams)? What patterns are used in your solution? What principle is being applied in your solution?

(e) Now, draw one UML diagram showing your redesign. Feel free to add other packages, classes and interfaces/abstract classes, if necessary.

(f) Derive the value of I, A and D' for each package in your design. What is the average D' value of your code?



Scratch sheet

What ever your write on this paper will not be graded.