ISYS1083/1084 Object Oriented Software Design

Tutelab 3 Refactoring

1. Briefly explain why the following practices are considered bad?

(a) Commented out code
(b) Large classes
(c) Feature Envy
(d) Duplicated switch statements
(e) Misplaced Responsibilities
(f) Inappropriate Intimacy between classes
(g) Long parameter list

2. How can *Extract Method* be applied to avoid code replication in the code below?

```
public class CustomerNameChanger
{
     public void changeName(CustomerContext, context, String
customerID, String name)
      {
           Customer customer = context.getCustomer(customerID);
           If (customer == null)
               Throw new Exception ("Customer with ID  " + customerID
+ " is not found");
          customer.name = name;
     }
}

public class CustomerAddressChanger
{
     public void changeAddress(CustomerContext, context, String
customerID, String address)
      {
           Customer customer = context.getCustomer(customerID);
           If (customer == null)
               Throw new Exception ("Customer with ID  " + customerID
+ " is not found");
          customer.address = address;
     }
}
```

3. The Circle and Square classes contain common methods to draw and move all such objects. Draw a class diagram showing appropriate classes and interfaces by applying the Extract Interface refactoring that will allow all such objects to be moved or displayed polymorphically.

```
class Circle {
   public Circle(double x, double y, double radius) {
      …
   }
   public void draw(Graphics g) {
   }
   public void move(int x, int y) {
   }
}
class Square {
   public Square(double x, double y, double side) {
      …
   }
```

```
      public void draw(Graphics g) {
      }
      public void move(int x, int y) {
      }
   }
```

4. Rewrite the Stack class below using delegation

```
class MyStack extends Vector {
    public void push(Object element) {
       insertElementAt(element,0);
    }
    public Object pop() {
       Object result = firstElement();
       removeElementAt(0);
       return result;
    }
}
```

5. Locate your own Java code from a previous semester (e.g. assignment or exercise) that could benefit from refactoring.

(a) Specify sections of the code that could be refactored citing the reasons.
(b) Identify at least one specific (Fowler) refactoring technique that can be applied citing the reason.
(c) Refactor and rewrite the code according to your answer to b)
(d) Explain how the refactored code improves
    (i) Understanding
    (ii) Maintainability