Sample Short-Answer Questions

1. (4pts) For the following code show what the order of the output will be.

**public** **class** MidtermSuperClass {

**static** **int** *superInt* = *superIntMethod*();

**static** **int** superIntMethod(){

System.***out***.println("Initializing variable superInt");

**return** 1;

}

MidtermSuperClass(){

System.***out***.println("Running midtermsuperclass constructor");}

}

**public** **class** Main {

**public** **static** **void** main(String[] args) {

MidtermSuperClass mySuper = **new** MidtermSubClass();

}}

**public** **class** MidtermSubClass **extends** MidtermSuperClass{

**static** **int** *subInt* = *subIntMethod*();

**static** **int** subIntMethod(){

System.***out***.println("Initializing variable subInt");

**return** 2;

}

MidtermSubClass(){

System.***out***.println("Running midtermsubclass constructor");

}}

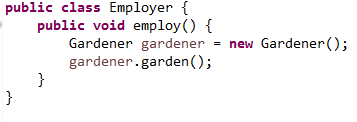
*Initializing variable superInt*

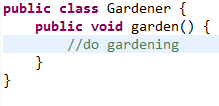
*Initializing variable subInt*

*Running midtermsuperclass constructor*

*Running midtermsubclass constructor*

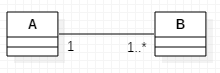
1. (2pts) In the following code, describe the relationship between Employer and Gardener.





*There is a dependency from Employer to Gardener*

1. (4pts) Describe the relationship between A and B from the following class diagram. Explain what must be in the code for each instance of A and B and explain any constraints on their constructors.



*Each instance of the class A contains a list of instances of B.*

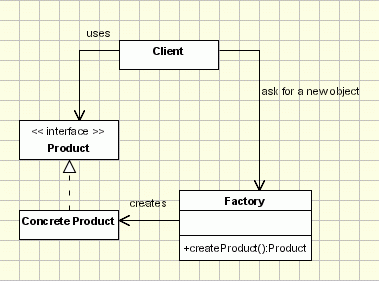
*Each instance of B contains exactly 1 reference to class A.*

*If an instance of A has been created, at least one instance of B has also been created. If an instance of B has been created one instance of A has also been created.*

1. (6pts) What is the OOD Open-Closed Principle? Write some simple code fragments showing a good example coding to the Open-Closed Principle for a problem where we compute the areas of shapes. Shapes can be circles, rectangles, or squares for this problem.

*Your code will show an application computing the area of these 3 types of shapes polymorphically. Circle, Rectangle and Square will be subclasses of Shape. Each subclass will override computeArea(). Shape should be abstract and computeArea() in shape should be an abstract method.*

1. (8pts)Show a UML class diagram for the Object Creation Factory Pattern. Create a simple example of this pattern and write enough code in each of your classes that shows the scenario of creating one concrete class. Just a few lines of code for each of your classes is enough to show your design.



Sample Code:

public class FactoryClient {

public static void main(String[] args) {

Product mtnBike;

ProductFactory myProductFactory = new ProductFactory();

mtnBike = myProductFactory.createProduct("MtnBike");

if (mtnBike != null)

System.out.println(mtnBike.getProductPrice());

}}

public class MtnBike implements Product {

@Override

public double getProductPrice() {

return 300.00;

}}

public interface Product {

public double getProductPrice();

}

public class ProductFactory {

public Product createProduct(String productType){

if (productType.equals("MtnBike"))

return new MtnBike();

else

return null;

}}

1. (2pts)What is the “diamond problem?” Draw a diagram showing the problem.

