Name: _____ CSSE 375 Midterm Exam Practice

1. (1 point) Which of the following is true of a well–executed refactoring?

   A. It relies on inheritance and polymorphism

   B. It has intermediate steps where the code is functional

   C. It uses a refactoring IDE like Eclipse

   D. It creates a new class (or maybe several)

   E. When the refactoring is finished, you usually need to add several unit tests to exercise the new functionality

2. (1 point) Which of the follow describes a Refused Bequest?

   A. NetworkedFile, a subclass of GameDataFile, that when you call saveToDisk() actually writes the file to a cloud storage across the network

   B. AutosaveRecord that has a variable NetworkHanlder that is usually null

   C. Manager, a subclass of employee, that returns -1 when the getEmployeeId() method is called because managers don't have employee ids

   D. LoginCommand, which duplicates many methods of NetworkCommand but is not Network-Commands' subclass

   E. CompositeWindow an abstract class designed to be a superclass that has no subclasses

3. (1 point) What smell does this source code suggest?

```
class Student
 {
  private String name;
  private int gradYear;
  private int studentID;

  public String getName() {...}
  public void setName(String name) {...}
  public int getGradYear() {...}
  public void setGradYear(int year) {...}
  public int getStudentID() {...}
  public void setStudentID(int id) {...}
}
```

   A. Data Class

   B. Short Class

   C. Refused Bequest

   D. Data Clumps

   E. Actually, this code is fine

4. (1 point) You and your friend are looking at the function signature below. Your fiends suggests that "this might be an instance of Primitive Obsession". What might your friend be proposing?

```
public int getCustomerForReferral(int customerIdOfReferrer, double referrerPercent, Stri
```

   A. That this function has a large number of parameters, many of them Java primitives, and that the number of parameters should be reduced

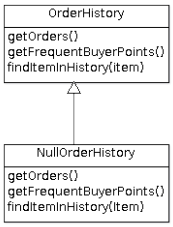
   B. That referrerPercent, url, and p could be combined into a single object

   C. That this method would make more sense if it was a method on the Product class

D. That the function would be improved if it used existing Java classes Integer and Double rather than int and double

E. That a new CustomerId object might be created, rather than using ints for customer ids

5. What smell does this UML diagram suggest?



A. Middle Man

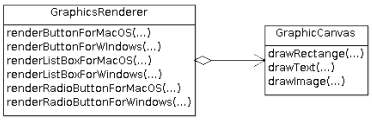
B. Data Clumps

C. Feature Envy

D. Parallel Inheritance Hierarchies

E. Actually, this code is fine

6. (1 point) Looking at this class diagram, what is the most significant design problem?



A. The GraphicsRenderer object is a Large Class and some methods should move to GraphicCanvas

B. The methods of GraphicsRenderer have Feature Envy and should move to GraphicCanvas

C. The methods of GraphicsRenderer indicate a Combinatorial Explosion

D. GraphicCanvas and GraphicRenderer should have a common superclass so they can Pull Up common code

E. Actually, this code is fine

7. (1 point) Oftentimes when you refactor a long method with several local variables, you end up with a lot of functions that take many parameters. Which of these is NOT a good potential solution for this problem?

A. Combine several of the parameters into a new object and pass that instead

B. Make methods that recompute the variables and call them from each of the functions that need them

C. Make a new Method Object that contains all the variables and functions related to this computation

D. Make the parameters instance variables of the class that are null unless the functions are called

E. All of the choices above are good potential solutions for this problem

8. (1 point) Which of these would be an example of divergent change?

A. A FormatParser class that needs to be subclassed in one way when you add a new add format, and another way when you add a new output type

B. A Sprite class in a video game and that you subclass every time you need a new kind of sprite and implement 3 different abstract methods

C. A HTTPProtocol class that has one gigantic method that every new feature needs to add to

D. A web system where you have to both update the C++ backend code as well as the Perl webpage code

E. A system where everytime you add a new DataElement class, you also need to add a new DataElementRenderer class

9. (1 point) Under what circumstances might you want to take two existing classes and give them a common superclass?

A. Both classes have similar methods and you can remove duplication by moving them to the superclass

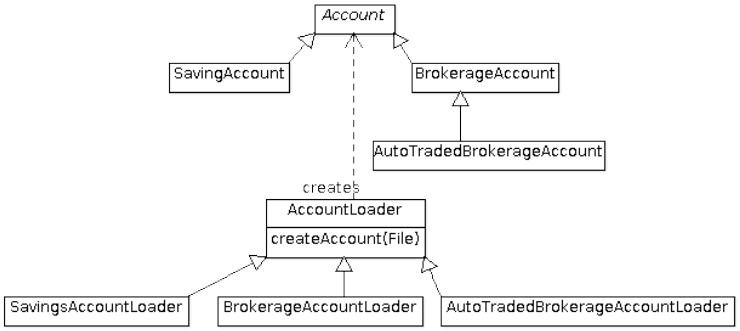
B. One class has several Temporary Variables that can be Pulled Up into the superclass

C. You need to use the superclass as a Middle Man for the clients of both of the classes

D. Both classes are part of Parallel Inheritance Hierarchies and want to remove the implicit duplcation by giving both hierarchies a shared interface

E. One class is a Large Class and moving methods into the superclass will make it smaller

10. (1 point) What smell does this diagram suggest?



A. Combinatorial Explosion

B. Middle Man

C. Refused Bequest

D. Parallel Inheritance Hierarchies

E. Actually, this code is fine

11. (1 point) You come across the following code. What refactoring would most improve it?

```
public void updateName(DataRecord newFile) {
  string result = null;
  while(newFile.hasNext()) {
    DataRecordEntry e = newFile.getNext();
    if(e.key().equals("name")) { result = e.value(); }
  }
  if(result == null) throw new RuntimeException("name not found");
  name = result;
}

public void updateName(DataRecord newFile) {
  string result = null;
```

```
  for(DataRecordEntry i = newFile.getNext(); newFile.hasNext(); i = newFile.getNext()) {
    if(i.key().equals("description")) { result = i.value(); }
  }
  if(result == null) throw new RuntimeException("description not found");
  description = result;
}
```

A. A single utility method should be extracted and called from both functions, eliminating the duplication

B. The variables e and i should be renamed to be more explanatory

C. A local variable should be introduced to explain the method's purpose more clearly

D. The functions should be changed to return an error code rather than throwing an exception when problems are found

E. i.key().equals(...) is a Message Chain and should be removed

12. (1 point) Object Oriented programmers often say that case statements are bad. Why?

A. Case statements encourage writing long methods

B. In languages like Java, strings cannot be used in case statements so they require you to use hard coded constants

C. Case statements are often vary behavior based on types, which can be replaced by polymorphism

D. Case statements introduce a strong performance overhead in OO languages because they can't be optimized the same way they can be in procedural languages like C

E. Case statements often have subtle bugs which more straightforward if statements don't

13. (1 point) Imagine you see the following code. What smell does it suggest?

```
class Person {
  public Person(DatabaseConnection c) {...}
  public Person(NetworkStore c) {...}

  public DatabaseConnection db; //should be null if person was initialized with net
  public NetworkStore net; //should be null if person was initialized with db
  ...
```

A. Primitive Obsession

B. Data Clumps

C. Divergent Change

D. Temporary Field

E. Actually, this code looks fine

14. (1 point) Which of these statements most nearly describes how Fowler thinks about comments in code?

A. You should have a comment in the header of every function, but not within the code itself

B. Comments are not a bad thing, but they can often be made unnecessary by clean code

C. Comments are useful in procedural languages like C, but not modern languages like Java

D. Comments are a necessary evil — we might want our code to be really clean but in real code you generally have to use a lot of comments

E. Whether or not you use comments is determined by your coding standards, and it's an aesthetic choice that has no effect on your code quality

15. (1 point) Imagine you see the following code. What smell does it suggest?

```
public boolean getsDiscount(Customer customer)
{
  if(customer.isSenior() || customer.isStudent())
    return true;
  if(customer.totalOrderCosts() > FREQUENT_CUSTOMER_CUTOFF)
    return true;
  return false;
}
```

A. Duplicate Code

B. Feature Envy

C. Middle Man

D. Temporary Field

E. Actually, this code is fine

16. (1 point) In an error reporting system, you notice a lot of the classes tend to have the same set of 3 instance variables: url, customerId, and timestamp. What might this suggest?

A. That these three variables often occur together, and should be replaced with a single identifier that links to a global map

B. That these three variables are a potential source of memory overhead, and you should profile your code to check

C. That these three objects might be extracted out into a single class

D. That your classes are likely repeating data and should be combined into one class

E. That there should be a common superclass of all the classes in the system, and these three fields should be protected members