NAME: _____

CSE403 Spring 2010 Midterm Examination
April 30, 2010

- 50 minutes.
- Open notes, open book.
- Closed any kind of electronics (including calculators and cell phones)
- Closed neighbor.
- Better legibility makes for happier graders.
- Make sure your name is on at least the first page

	Maximum Points	Earned Points
Q1	10	
Q2	10	
Q3	15	
Q4	10	
Q5	20	
Q6	15	
Total	80	

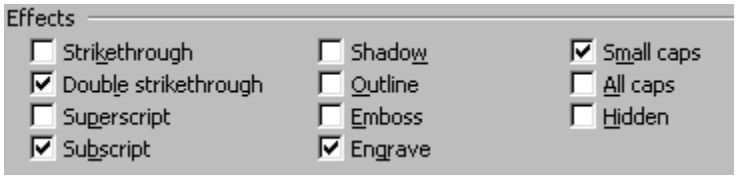
NAME: _____

Q1 [10 points] A common way to distinguish requirements from design (or implementation) is to characterize requirements as “the what” and design as “the how.” Concisely give *one* reason why this characterization is insufficient.

Q2 [10 points] Consider the two projects for the course: the co-authorship tool and the clone detection tool. Which of the projects would be a better match to a waterfall lifecycle model? Briefly justify your answer.

NAME: _____

Q3 [15 points] Consider the following element from a conventional Microsoft text formatting box. Briefly identify a weakness of the interface, particularly focusing on the degree to which the effects are or are not independent of each other. Recommend a way to overcome the weakness.



NAME: _____

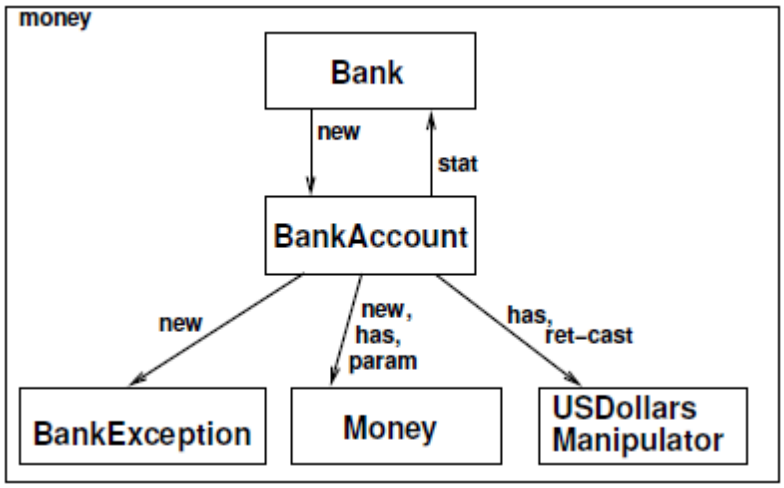
Q4 [10 points] Consider the following Java code

```
public class Money {
    private double m_amount;
    private String m_currency_type; // "USDollar", "Euro", etc.

    public Money(double amount) {
        m_amount = amount;
        m_currency_type = Globals.getCurrencyType();
    }

    public double getAmount() {
        return m_amount;
    }

    public String getCurrencyType() {
        return m_currency_type;
    }
}
```

What design decisions (specifically, representations of data) – if any – are encapsulated by the **Money** class? That is, what decisions could be changed without causing the clients of the class to be modified?

NAME: _____

Q5 [20 points] Consider the following dependency diagram that includes this **`Money`** class (ignore **`BankException`**):



Briefly, the diagram says that:

- The **`Bank`** class creates new **`BankAccount`** instances, which can in turn return their status to **`Bank`** instances.
- The **`BankAccount`** class creates new instances of **`Money`**, includes instances of **`Money`**, and passes instance of **`Money`** as parameters.
- The **`BankAccount`** class includes instances of **`USDollarsManipulator`** and performs casts to **`USDollarsManipulator`** on values returned from **`USDollarsManipulator.`**

I claim that this design makes it difficult to change **`BankAccount`** to allow instances to manage multiple currencies (say, **`USDollars`** and **`Euros`**).

Either: (a) argue concisely that my claim is inaccurate; *or* (b) sketch a different structure that would make it easier to apply this change.

NAME: _____

Q6 [15 points total, 5 points each]

1. True or false: When a subclass inherits from a superclass, no additional coupling is created in the design. Briefly justify your answer (1-2 sentences).

2. True or false: Using a formal logic cannot overcome all problems in software requirements specification. Briefly justify your answer (1-2 sentences).

3. True or false: There is a close relationship between use cases and UML sequence diagrams. Briefly justify your answer (1-2 sentences).