Problem 1 [4pt] Consider the following Java code:

```java
abstract class A {
    private A f;

    public A() { } // constructor 1
    public A(A foo) { this.f = foo; } // constructor 2

    public A getF() { return this.f; }
    public abstract A m1();
    public void m2() {
        A temp;
        temp = this.m1();
        m3(temp);
    }
    public static void m3(A a) { a.m1(); }
}
class B extends A {
    public B() { super(); } // constructor 1
    public B(A bar) { super(bar); } // constructor 2
    public A m1() { return this; }
}
class C extends B {
    public C(A baz) { super(baz); }
    public A m1() { return super.m1().getF(); }
}
```

1. Write the equivalent C++ code. If you want to ensure that you have correct C++ code, try something like “g++ -c myfile.cpp” on stdsun, where myfile.cpp contains the code. Do not worry about deallocation of objects (i.e., you do not have to use `delete` in your C++ code to free up the memory created by `new`).

2. Consider the following Java code that uses the three classes from above:

```java
A v1, v2;
// moment 1
v1 = new B();
v2 = new C(v1);
v2.m2();
// moment 2
```

List the sequence of call and return events from moment 1 to moment 2. The list should look like this:

- Constructor `B()` in class `B` is invoked (first event)
- ...
- The call to `B()` returns
- Constructor `C(C baz)` in class `C` is invoked
- ...
- The call to `C(C baz)` returns
- Method `m2()` in class `A` is invoked
- ...
- The call to `m2()` returns (last event)

Fill in the blanks with the rest of the events. Describe precisely which method in which class is called.
IMPORTANT: If you do not know Java, you must make sure that you completely understand the meaning of this code. In class we discussed the key features of the language. If you are used to C++, keep in mind that the declaration “A f” from above is essentially equivalent to a declaration “A *f” in C++. In other words, the field in class A is a pointer to objects. Similarly, the formal parameters of the constructors and the methods are pointers to objects; the same is also true for temp, v1, and v2. If you have any doubts about your understanding of this code, discuss them with me. Lack of previous experience with Java will not be accepted as an excuse for doing poorly on this question. I will definitely ask simple Java-based questions in the midterm and the final; problem 3 below is a typical example of what you should expect to see in the exams.

Problem 2 [3pt]

Consider this UML collaboration diagram. Draw an equivalent UML sequence diagram. Make sure that you include the appropriate object lifelines and method activation boxes, as well as all conditions in the appropriate places. To simplify the diagram, do not show any return edges.

Problem 3 [3pt] Consider the following Java classes:

```java
class CD {
    private Artist artist;
    private Date date;
    // ----- constructor
    public CD (Artist a) {
        this.artist = a;
        this.date = new Date();
    }
    // ----- short string description
    public String shortDescr() {
        return ("Artist: " + this.artist.getName());
    }
    // ----- longer string description
    public String longDescr() {
        return (this.shortDescr() + " Year: " + this.date.getYear());
    }
}
class Artist {
    ... // some code
    public String getName() { ... // some code }
}
class Date {
    ... // some code
    public Date() { ... // some code }
    public int getYear() { ... // some code }
}
```
IMPORTANT: Again, if you do not know Java, you must make sure that you completely understand the meaning of this code. Talk to me about every aspect of the code that is not clear to you.

1. Consider the creation of a new object of class CD by some expression of the form “new CD(x)”, where x is a pointer to some existing object of class Artist. The evaluation of this expression has two steps: (1) creation of a new object, and (2) invocation of the constructor of class CD on this new object. We can represent these two steps using a sequence diagram. The starting point of the diagram is a create message sent to an instance of CD. The rest of the diagram shows all messages (i.e., method calls) that happen from the moment of time when the constructor is entered to the moment of time when the constructor completes, in the appropriate order. For each message, the parameters should be shown using the UML notation described in class. Draw the sequence diagram.

2. Consider the expression “x.longDescr()” where x points to some object of class CD. The evaluation of this expression invokes method longDescr on this object. Show a collaboration diagram that represents this invocation. The starting point of the diagram is a message longDescr being sent to an instance of CD. The rest of the diagram should show all messages that happen from the moment of time when longDescr is entered to the moment of time when longDescr completes, in the appropriate order.