1. Consider the abstract class Politician and two concrete subclasses, Senator and President. All Politicians implement Corruptible:

```java
interface Corruptible {
    void acceptMoney(int amount); // bribe money
    void setQuote(String s); // words the Politician is paid to say
    String getQuote(); // given to Reporters
}
```

All Politicians accept Visitors, including subclasses: Lobbyist and Reporter.

When a Lobbyist visits a Senator, the Senator accepts bribe money and is told what to say to subsequent Reporters. When a Lobbyist visits a President the two are caught on camera (see methods next page for details).

A Reporter visiting either a Senator or President just gets a quote which is subsequently printed.

For instance:

```java
Politician ted = new Senator("Ted");
Politician george = new President("George");
Lobbyist jack = new Lobbyist("Jack");
Reporter bob = new Reporter("Bob");

jack.setBribe(10000); // give lobbyist money to pass to politician
jack.setQuote("we must drill in Anwar"); // tell politicians what to say
ted.acceptVisitor(jack); // send the lobbyist to Senator Ted
ted.acceptVisitor(bob); // Reporter Bob visits, gets quote from Ted
bob.printQuote(); // Ted’s words then get printed, and
george.acceptVisitor(jack); // George caught photographed with lobbyist
```

results in the output:

An unnamed source said we must drill in Anwar
George photographed with Jack, Click!

The code is provided in part on the next two pages. You are to compete the code so it would compile and run correctly.
1a) [25%] Complete this code for the Politician hierarchy (fill in ALL missing code, some requiring that you complete a line of Java). Note that Politicians do nothing between visits (they have no other methods).

```java
class Politician
private String quote;
private int    funds;
private String name;

Politician(String name) { this.name = name; }

public String getName() { return name; }

public void bePhotographedWith(Visitor v) {
    System.err.println(name + " photographed with " + v.getName() + ", Click!");
}

public void acceptMoney(int amount) { funds += amount; }
public void setQuote(String s)      { quote = s; }
public String getQuote()              { return quote;}

class Senator
    Senator(String name)

class President
    President(String name)
```
1b) [25%] And regarding the Visitor hierarchy:

```java
class Visitor {
    private String name;

    Visitor(String name) { this.name = name; }
    public String getName() { return name; }
}
```

```java
class Lobbyist {
    private int bribe = 0;
    private String quote = "";

    Lobbyist(String name) {
    }
    void setBribe(int x) { bribe = x; }
    void setQuote(String s) { quote = s; }
}
```

```java
class Reporter {
    private String quote = "";

    Reporter(String name) {
    }
    void printQuote() { System.err.println("An unnamed source said "+ quote);
}
```
2) [10%] Draw a UML class diagram for the three classes of the Visitor hierarchy, being sure to indicate abstract classes and methods.

3) [10%] Draw a UML sequence diagram to show the flow of control for the following (from main). Assume jack and ted are valid instances of Lobbyist and Senator. Show method calls and the returns from method calls.

```java
jack.setBribe(10000); // give lobbyist money to pass to politician
jack.setQuote("we must drill in Anwar"); // tell politicians what to say
fred.acceptVisitor(jack); // send the lobbyist to Senator Ted
```

```
main

jack: Lobbyist

fred: Senator
```
4) [30%] Given:

```java
interface I {
    public method m1();
}

abstract class A implements I {
    public method m1() {
        System.out.println("m1 in A");
    }
}

class B extends A {
    public method m1() {
        System.out.println("m1 in B");
    }
    public method m2() {
        System.out.println("m2 in B");
    }
}
```

```java
B b = new B();
A a = b;
I i = b;
```

a) [5%] Write an expression to call method m2 using variable a.

b) [5%] Write an expression to call method m2 using variable i.

c) [5%] What information is lost when one assigns an instance of a B to a variable of type I, and then assigns it back to a different variable of type B? Example:

```java
B b1 = new B();
I i = (I)b;
B b2 = (B)i;
```

d) [10%] What are two basic uses of interfaces?

e) [5%] What are two reasons for making a class abstract?