1. (2 pts) What is printed when the class Test is executed? Explain the output.

   ```java
   public class Question1 {
       public void print() {
           System.out.println("Question 1");
       }
       public static void print1() {
           System.out.println("Question 1");
       }
   }

   class Test extends Question1 {
       public void print() {
           System.out.println("Test");
       }
       public static void print1() {
           System.out.println("Test");
       }
       public static void main(String[] args) {
           Question1 question1 = new Test();
           Test test = new Test();
           question1.print();
           test.print();
           question1.print1();
           test.print1();
       }
   }
   ```

   The correct instance method to invoke is determined by the runtime type of the caller. The correct class method to invoke is determined by the compile-time type of the reference.

2. (2 pts) The keyword `protected` allows entities to be accessed directly by subclasses even if they are not in the same package as their superclass.

3. (2 pts) Explain the two errors in the following code.

   ```java
   public final class Question3 {
       public final void print() {
           System.out.println("Question 3");
       }
   }

   class Test1 extends Question3 {
       public void print() {
           System.out.println("Test 1");
       }
   }
   ```

   A final class cannot be extended. A final instance method cannot be overridden.

4. (2 pts) **Polymorphism** refers to a dynamic binding that occurs at runtime between a method call and implementation. This only applies to instance methods since if we implement a class method with the same signature and return type as one in the superclass, then the one in the superclass is hidden in the subclass.

5. (2 pts) Suppose that class A extends B and class B extends C. Which of the following are valid declarations? Explain. Assume that no constructors are defined in the classes.
```java
A a = new B();
B b = new A();
C c = new A();
B b = new C();
A a = new C();
```

B b = new A() and C c = new A() are valid since A is a subclass of B and A is a subclass of C. The others are invalid because instances of a superclass are not necessarily instances of a subclass.