Question 12

A class can not be called "tightly encapsulated" unless which of the following are true?

a. The data members can not be directly manipulated by external code.
b. The class is declared final.
c. It has no public mutator methods.
d. The superclass is tightly encapsulated.

Question 13

```
class A {
  String s1 = "A.s1"; String s2 = "A.s2";
}
class B extends A {
  String s1 = "B.s1";
  public static void main(String args[]) {
    B x = new B(); A y = (A)x;
    System.out.println(x.s1+" "+x.s2+" "+y.s1+" "+y.s2);
}}
```

What is the result of attempting to compile and run the program?

a. Prints: B.s1 A.s2 B.s1 A.s2
b. Prints: B.s1 A.s2 A.s1 A.s2
c. Prints: A.s1 A.s2 B.s1 A.s2
d. Prints: A.s1 A.s2 A.s1 A.s2
e. Run-time error
f. Compile-time error
g. None of the above

Question 14

```
class A {void m1(A a) {System.out.print("A");}}
class B extends A {void m1(B b) {System.out.print("B");}}
class C extends B {void m1(C c) {System.out.print("C");}}
class D extends C {
  void m1(D d) {System.out.print("D");}
  public static void main(String[] args) {
    A a1 = new A(); B b1 = new B();
    C c1 = new C(); D d1 = new D();
    d1.m1(a1); d1.m1(b1); d1.m1(c1);
}}
```

What is the result of attempting to compile and run the program?

a. Prints: AAA
b. Prints: ABC
c. Prints: DDD
d. Prints: ABCD
e. Compile-time error
f. Run-time error
g. None of the above

Question 15

```
class C {
  void printS1() {System.out.print("C.printS1 ");}
  static void printS2() {System.out.print("C.printS2 ");}
}
class D extends C {
  void printS1(){System.out.print("D.printS1 ");}
  void printS2() {System.out.print("D.printS2 ");}
  public static void main (String args[]) {
    C c = new D(); c.printS1(); c.printS2();
}}
```

What is the result of attempting to compile and run the program?

a. Prints: C.printS1 C.printS2
b. Prints: C.printS1 D.printS2
c. Prints: D.printS1 C.printS2
d. Prints: D.printS1 D.printS2
e. Run-time error
f. Compile-time error
g. None of the above

Question 16

```
class A {}
class B extends A {}
class C extends B {}
class D {
  void m1(A a) {System.out.print("A");}
  void m1(B b) {System.out.print("B");}
  void m1(C c) {System.out.print("C");}
  public static void main(String[] args) {
    A c1 = new C(); B c2 = new C();
    C c3 = new C(); D d1 = new D();
```

```
    d1.m1(c1); d1.m1(c2); d1.m1(c3);
}}
```

What is the result of attempting to compile and run the program?

a. Prints: AAA
b. Prints: ABC
c. Prints: CCC
d. Compile-time error
e. Run-time error
f. None of the above

Question 17

```
class E {
  void printS1(){System.out.print("E.printS1 ");}
  static void printS2() {System.out.print("E.printS2");}
}
class F extends E {
  void printS1(){System.out.print("F.printS1 ");}
  static void printS2() {System.out.print("F.printS2");}
  public static void main (String args[]) {
    E x = new F(); x.printS1(); x.printS2();
}}
```

What is the result of attempting to compile and run the program?

a. Prints: E.printS1 E.printS2
b. Prints: E.printS1 F.printS2
c. Prints: F.printS1 E.printS2
d. Prints: F.printS1 F.printS2
e. Run-time error
f. Compile-time error
g. None of the above

Question 18

```
class A {void m1(A a) {System.out.print("A");}}
class B extends A {void m1(B b) {System.out.print("B");}}
class C extends B {void m1(C c) {System.out.print("C");}}
class D {
  public static void main(String[] args) {
    A c1 = new C(); B c2 = new C();
    C c3 = new C(); C c4 = new C();
```

```
    c4.m1(c1); c4.m1(c2); c4.m1(c3);
}}
```

What is the result of attempting to compile and run the program?

a. Prints: AAA
b. Prints: ABC
c. Prints: CCC
d. Compile-time error
e. Run-time error
f. None of the above

Question 19

```
class P {
  static void printS1(){System.out.print("P.printS1 ");}
  void printS2() {System.out.print("P.printS2 ");}
  void printS1S2(){printS1();printS2();}
}
class Q extends P {
  static void printS1(){System.out.print("Q.printS1 ");}
  void printS2(){System.out.print("Q.printS2 ");}
  public static void main(String[] args) {
    new Q().printS1S2();
}}
```

What is the result of attempting to compile and run the program?

a. Prints: P.printS1 P.printS2
b. Prints: P.printS1 Q.printS2
c. Prints: Q.printS1 P.printS2
d. Prints: Q.printS1 Q.printS2
e. Run-time error
f. Compile-time error
g. None of the above

Question 20

```
class A {void m1(A a) {System.out.print("A");}}
class B extends A {void m1(B b) {System.out.print("B");}}
class C extends B {void m1(C c) {System.out.print("C");}}
class D {
  public static void main(String[] args) {
    A c1 = new C(); C c2 = new C(); c1.m1(c2);
}}
```

What is the result of attempting to compile and run the program?

a. Prints: A
b. Prints: B
c. Prints: C
d. Compile-time error
e. Run-time error
f. None of the above

Question 21

```
class R {
  private void printS1(){System.out.print("R.printS1 ");}
  protected void printS2() {System.out.print("R.printS2 ");}
  protected void printS1S2(){printS1();printS2();}
}
class S extends R {
  private void printS1(){System.out.print("S.printS1 ");}
  protected void printS2(){System.out.print("S.printS2 ");}
  public static void main(String[] args) {
    new S().printS1S2();
}}
```

What is the result of attempting to compile and run the program?

a. Prints: R.printS1 R.printS2
b. Prints: R.printS1 S.printS2
c. Prints: S.printS1 R.printS2
d. Prints: S.printS1 S.printS2
e. Run-time error
f. Compile-time error
g. None of the above

Question 22

```
class A {void m1(A a) {System.out.print("A");}}
class B extends A {void m1(B b) {System.out.print("B");}}
class C extends B {void m1(C c) {System.out.print("C");}}
class D {
  public static void main(String[] args) {
    A a1 = new A(); A b1 = new B();
    A c1 = new C(); C c4 = new C();
    a1.m1(c4); b1.m1(c4); c1.m1(c4);
}}
```

What is the result of attempting to compile and run the program?

a. Prints: AAA
b. Prints: ABC
c. Prints: CCC
d. Compile-time error
e. Run-time error
f. None of the above

12	a d	The data members can not be directly manipulated by external code. The superclass is tightly encapsulated.	If a class `A` is not tightly encapsulated, then no subclass of `A` is tightly encapsulated.
13	b	Prints: B.s1 A.s2 A.s1 A.s2	The variables of a subclass can hide the variables of a superclass or interface. The variable that is accessed is determined at compile-time based on the type of the reference--not the run-time type of the object. The two references `x` and `y` refer to the same instance of type `B`. The name `x.s1` uses a reference of type `B`; so it refers to the variable `s1` declared in class `B`. The name `y.s1` uses a reference of type `A`; so it refers to the variable `s1` declared in class `A`.
14	b	Prints: ABC	The method invocation expression `d1.m1(a1)` uses reference `d1` of type `D` to invoke method `m1`. Since the reference `d1` is of type `D`, the class `D` is searched for an applicable implementation of `m1`. The methods inherited from the superclasses, `C`, `B` and `A`, are included in the search. The argument, `a1`, is a variable declared with the type `A`; so method `A.m1(A a)` is invoked.
15	f	Compile-time error	Suppose a superclass method is not private and is accessible to code in a subclass. If the superclass method is declared `static`, then any subclass method sharing the same signature must also be declared `static`. Similarly, if the superclass method is declared non-static, then any subclass method sharing the same signature must also be declared non-static. The attempted declaration of the non-static method `D.printS2` generates a compile-time error; because the superclass method, `C.printS2`, is `static`.
16	b	Prints: ABC	Three methods overload the method name `m1`. Each has a single parameter of type `A` or `B` or `C`. For any method invocation expression of the form `m1(referenceArgument)`, the method is selected based on the declared type of the variable `referenceArgument`--not the run-time type of the referenced object. The method invocation expression `d1.m1(c1)` uses reference `d1` of type `D` to invoke method `m1` on an instance of type `D`. The argument, `c1`, is a reference of type `A` and the run-time type of the referenced object is `C`. The argument type is determined by the declared type of the reference variable `c1`--not the run-time type of the object referenced by `c1`. The declared type of `c1` is type `A`; so the method `m1(A a)` is selected. The declared type of `c2` is type `B`; so the method invocation expression `d1.m1(c2)` invokes method `m1(B b)`. The declared type of `c3` is type `C`; so the method invocation expression `d1.m1(c3)` invokes method `m1(C c)`.
17	c	Prints: F.printS1 E.printS2	A static method is selected based on the compile-time type of the reference--not the run-time type of the object. A non-static method is selected based on the

			run-time type of the object--not the compile-time type of the reference. Both method invocation expressions, `x.printS1()` and `x.printS2()`, use a reference of the superclass type, `E`, but the object is of the subclass type, `F`. The first of the two expressions invokes an instance method on an object of the subclass type; so the overriding subclass method is selected. The second invokes a static method using a reference of the superclass type; so the superclass method is selected.
18	b	Prints: ABC	Three methods overload the method name `m1`. Each has a single parameter of type `A` or `B` or `C`. For any method invocation expression of the form `m1(referenceArgument)`, the method is selected based on the declared type of the variable `referenceArgument`--not the run-time type of the referenced object. The method invocation expression `c4.m1(c1)` uses reference `c4` of type `C` to invoke method `m1` on an instance of type `C`. The argument, `c1`, is a reference of type `A` and the run-time type of the referenced object is `C`. The argument type is determined by the declared type of the reference variable `c1`--not the run-time type of the object referenced by `c1`. The declared type of `c1` is type `A`; so the method `A.m1(A a)` is selected. The declared type of `c2` is type `B`; so the method invocation expression `c4.m1(c2)` invokes method `B.m1(B b)`. The declared type of `c3` is type `C`; so the method invocation expression `c4.m1(c3)` invokes method `C.m1(C c)`.
19	b	Prints: P.printS1 Q.printS2	Suppose a method `m1` is invoked using the method invocation expression `m1()`. If `m1` is a `static` member of the class where the invocation expression occurs, then that is the implementation of the method that is invoked at run-time regardless of the run-time type of the object. If `m1` is non-static, then the selected implementation is determined at run-time based on the run-time type of the object. The program invokes method `printS1S2` on an instance of class `Q`. The body of method `printS1S2` contains two method invocation expressions, `printS1()` and `printS2()`. Since method `printS1` is static, the implementation declared in class `P` is invoked. Since `printS2` is non-static and the run-time type of the object is `Q`, the invoked method is the one declared in class `Q`.
20	a	Prints: A	The reference `c1` is of the superclass type, `A`; so it can be used to invoke only the method `m1` declared in class `A`. The methods that overload the method name `m1` in the subclasses, `B` and `C`, can not be invoked using the reference `c1`. A method invocation conversion promotes the argument referenced by `c2` from type `C` to type `A`, and the method declared in class `A` is executed. Class `A` declares only one method, `m1`. The single parameter is of type `A`. Class `B` inherits the method declared in class `A` and overloads the method name with a new method that has a single parameter of type `B`. Both methods sharing the overloaded name, `m1`, can be invoked using a reference of type `B`; however, a reference of type `A` can be used to invoke only the method declared in class `A`. Class `C` inherits the methods declared in classes `A` and `B` and overloads the method name with a new method that has a single parameter of type `C`. All three methods sharing the overloaded name, `m1`, can be invoked using a reference of type `C`; however, a reference of type `B` can be used to invoke only the method declared in class `B` and the method declared in the superclass `A`. The method invocation expression `c1.m1(c2)` uses reference `c1` of type `A` to invoke method `m1`. Since the reference `c1` is of type `A`, the search for an applicable

			implementation of `m1` is limited to class `A`. The subclasses, `B` and `C`, will not be searched; so the overloading methods declared in the subclasses can not be invoked using a reference of the superclass type.
21	b	Prints: R.printS1 S.printS2	A private method of a superclass is not inherited by a subclass. Even if a subclass method has the same signature as a superclass method, the subclass method does not override the superclass method. Suppose a non-static method `m1` is invoked using the method invocation expression `m1()`. If `m1` is a `private` member of the class `T` where the invocation expression occurs, then the implementation in class `T` is selected at run-time regardless of the run-time type of the object. If the non-static method `m1` is not `private`, then the selected implementation is determined at run-time based on the run-time type of the object. The program invokes the non-static method `printS1S2` on an instance of class `S`, so the run-time type is `S`. The body of method `R.printS1S2` contains two method invocation expressions, `printS1()` and `printS2()`. Since class `R` contains a `private` implementation of the instance method `printS1`, it is the implementation that is selected regardless of the run-time type of the object. Since `printS2` is not `private` and not `static`, the selected implementation of `printS2` depends on the run-time type of the object. The method `printS1S2` is invoked on an instance of class `S`; so the run-time type of the object is `S`, and the implementation of `printS2` declared in class `S` is selected.
22	a	Prints: AAA	The declared type of the reference variables, `a1`, `b1` and `c1`, is the superclass type, `A`; so the three reference variables can be used to invoke only the method `m1(A a)` that is declared in the superclass, `A`. The methods that overload the method name `m1` in the subclasses, `B` and `C`, can not be invoked using a reference variable of the superclass type, `A`. A method invocation conversion promotes the argument referenced by `c4` from type `C` to type `A`, and the method declared in class `A` is executed.