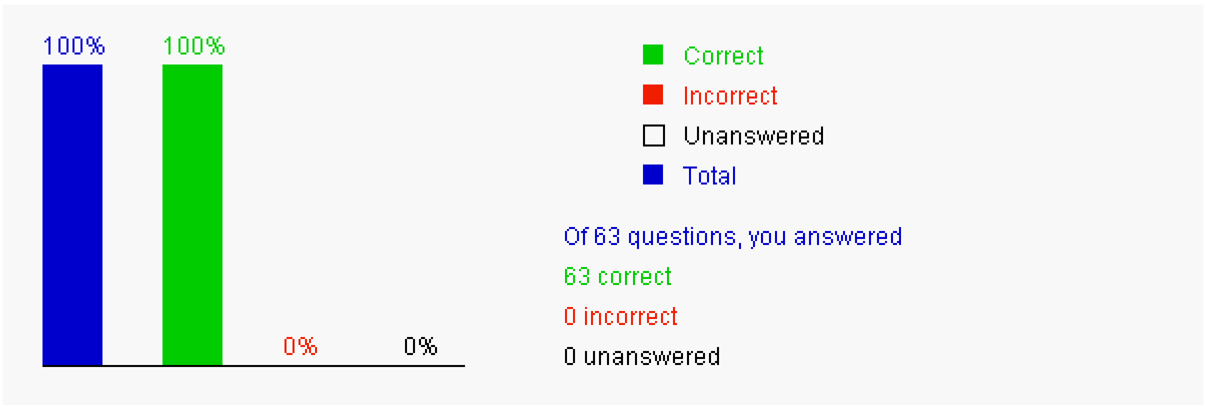
Introduction to Java Programming, Includes Data Structures, Eleventh Edition, Y. Daniel Liang

This quiz is for students to practice. A large number of additional quiz is available for instructors using Quiz Generator from the Instructor's Resource Website. Videos for Java, Python, and C++ can be found at https://yongdanielliang.github.io/revelvideos.html.

Chapter 11 Inheritance and Polymorphism



Of 63 questions, you answered
63 correct
0 incorrect
0 unanswered

Please send suggestions and errata to Dr. Liang at y.daniel.liang@gmail.com. Indicate which book and edition you are using. Thanks!

Section 11.2 Superclasses and Subclasses

11.1 Object-oriented programming allows you to derive new classes from existing classes. This is called ____________.

- A. encapsulation
- [✓ B. inheritance]
- C. abstraction
- D. generalization

Your answer is correct

11.2 Which of the following statements are true?

- A. A subclass is a subset of a superclass.
- [✓ B. A subclass is usually extended to contain more functions and more detailed information than its superclass.]
- [✓ C. "class A extends B" means A is a subclass of B.]
- D. "class A extends B" means B is a subclass of A.

Your answer is correct

Section 11.3 Using the super Keyword Section 11.3.1 Calling Superclass Constructors

11.3 Suppose you create a class Square to be a subclass of GeometricObject. Analyze the following code:

```
class Square extends GeometricObject {
  double length;

  Square(double length) {
    GeometricObject(length);
  }
}
```

- A. The program compiles fine, but you cannot create an instance of Square because the constructor does not specify the length of the Square.
- [✓ B. The program has a compile error because you attempted to invoke the GeometricObject class's constructor illegally.]
- C. The program compiles fine, but it has a runtime error because of invoking the Square class's constructor illegally.

Your answer is correct
Explanation: You have to use super() or super(withapproriatearguments) to invoke a super class constructor explicitly.

11.4 Analyze the following code:

```
public class A extends B {
}

class B {
  public B(String s) {
  }
}
```

- A. The program has a compile error because A does not have a default constructor.
- [✓ B. The program has a compile error because the default constructor of A invokes the default constructor of B, but B does not have a default constructor.]
- C. The program would compile fine if you add the following constructor into A: A(String s) { }
- [✓ D. The program would compile fine if you add the following constructor into A: A(String s) { super(s); }]

Your answer is correct

11.5 Analyze the following code:

```
public class Test extends A {
  public static void main(String[] args) {
    Test t = new Test();
    t.print();
  }
}

class A {
  String s;

  A(String s) {
    this.s = s;
  }

  public void print() {
    System.out.println(s);
  }
}
```

- A. The program does not compile because Test does not have a default constructor Test().
- [✓ B.] The program has an implicit default constructor Test(), but it cannot be compiled, because its super class does not have a default constructor. The program would compile if the constructor in the class A were removed.
- [✓ C.] The program would compile if a default constructor A(){ } is added to class A explicitly.
- D. The program compiles, but it has a runtime error due to the conflict on the method name print.

Your answer is correct
Explanation: See the last Note in the section, 'Using the super keyword.'

Section 11.3.2 Constructor Chaining

11.6 What is the output of running class C?

```
class A {
  public A() {
    System.out.println(
      "The default constructor of A is invoked");
  }
}

class B extends A {
  public B() {
    System.out.println(
      "The default constructor of B is invoked");
  }
}

public class C  {
  public static void main(String[] args) {
    B b = new B();
  }
}
```

- A. Nothing displayed
- B. "The default constructor of B is invoked"
- [✓ C.] "The default constructor of A is invoked" followed by "The default constructor of B is invoked"
- D. "The default constructor of B is invoked" followed by "The default constructor of A is invoked"
- E. "The default constructor of A is invoked"

Your answer is correct
Explanation: Superclass's constructor is called before executing the statements in the subclass constructor.

11.7 Which of the following is incorrect?

- [✓ A.] A constructor may be static.
- B. A constructor may be private.
- C. A constructor may invoke a static method.
- D. A constructor may invoke an overloaded constructor.
- E. A constructor invokes its superclass no-arg constructor by default if a constructor does not invoke an overloaded constructor or its superclass's constructor.

Your answer is correct
Explanation: A constructor cannot be static, because you use a constructor to create a specific instance. A constructor may be private. In this case, the use cannot create an instance using this constructor. For example, the constructor in the Math class is private. A constructor may invoke a static method

just like any method can invoke a static method. A constructor can invoke an overloaded constructor using the this keyword. So, the correct answer is A.

Section 11.3.3 Calling Superclass Methods

11.8 Which of the statements regarding the super keyword is incorrect?

- ○ A. You can use super to invoke a super class constructor.
- ○ B. You can use super to invoke a super class method.
- ● C. You can use super.super.p to invoke a method in superclass's parent class.
- ○ D. You cannot invoke a method in superclass's parent class.

Your answer is correct ✓
Explanation: Using super.super is not allowed in Java. So, the answer to this question is C.

Section 11.4 Overriding Methods

11.9 Analyze the following code:

```java
public class Test {
  public static void main(String[] args) {
    B b = new B();
    b.m(5);
    System.out.println("i is " + b.i);
  }
}

class A {
  int i;

  public void m(int i) {
    this.i = i;
  }
}

class B extends A {
  public void m(String s) {
  }
}
```

- ○ A. The program has a compile error, because m is overridden with a different signature in B.
- ○ B. The program has a compile error, because b.m(5) cannot be invoked since the method m(int) is hidden in B.
- ○ C. The program has a runtime error on b.i, because i is not accessible from b.
- ● D. The method m is not overridden in B. B inherits the method m from A and defines an overloaded method m in B.

Your answer is correct ✓

11.10 The getValue() method is overridden in two ways. Which one is correct?

```java
I:
public class Test {
  public static void main(String[] args) {
    A a = new A();
    System.out.println(a.getValue());
  }
}

class B {
  public String getValue() {
    return "Any object";
  }
}

class A extends B {
  public Object getValue() {
    return "A string";
  }
}

II:
public class Test {
  public static void main(String[] args) {
    A a = new A();
    System.out.println(a.getValue());
  }
}

class B {
  public Object getValue() {
    return "Any object";
  }
}

class A extends B {
  public String getValue() {
    return "A string";
  }
}
```

A. I

[✓ B. II]

C. Both I and II

D. Neither

Your answer is correct

Section 11.5 Overriding vs. Overloading

11.11 Which of the following statements are true?

[✓ A. To override a method, the method must be defined in the subclass using the same signature and compatible return type as in its superclass.]

[✓ B. Overloading a method is to provide more than one method with the same name but with different signatures to distinguish them.]

[✓ C. It is a compile error if two methods differ only in return type in the same class.]

[✓ D. A private method cannot be overridden. If a method defined in a subclass is private in its superclass, the two methods are completely unrelated.]

[✓ E. A static method cannot be overridden. If a static method defined in the superclass is redefined in a subclass, the method defined in the superclass is hidden.]

Your answer is correct

11.12 Which of the following statements are true?

[✓ A. A method can be overloaded in the same class.]

B. A method can be overridden in the same class.

C. If a method overloads another method, these two methods must have the same signature.

[✓ D. If a method overrides another method, these two methods must have the same signature.]

[✓ E. A method in a subclass can overload a method in the superclass.]

Your answer is correct

11.13 Analyze the following code:

```
public class Test {
  public static void main(String[] args) {
    new B();
  }
}

class A {
  int i = 7;

  public A() {
    System.out.println("i from A is " + i);
  }

  public void setI(int i) {
    this.i = 2 * i;
  }
}

class B extends A {
  public B() {
    setI(20);
    // System.out.println("i from B is " + i);
  }

  @Override
  public void setI(int i) {
    this.i = 3 * i;
  }
}
```

A. The constructor of class A is not called.

[✓ B. The constructor of class A is called and it displays "i from A is 7".]

C. The constructor of class A is called and it displays "i from A is 40".

D. The constructor of class A is called and it displays "i from A is 60".

Your answer is correct

Explanation: When invoking new B(), B's superclass A's constructor is invoked first. So it displays i from A is 7.

11.14 Analyze the following code:

```java
public class Test {
  public static void main(String[] args) {
    new B();
  }
}

class A {
  int i = 7;

  public A() {
    setI(20);
    System.out.println("i from A is " + i);
  }

  public void setI(int i) {
    this.i = 2 * i;
  }
}

class B extends A {
  public B() {
    // System.out.println("i from B is " + i);
  }

  @Override
  public void setI(int i) {
    this.i = 3 * i;
  }
}
```

A. The constructor of class A is not called.

B. The constructor of class A is called and it displays "i from A is 7".

C. The constructor of class A is called and it displays "i from A is 40".

[✓ D. The constructor of class A is called and it displays "i from A is 60".]

Your answer is correct

Explanation: When invoking new B(), B's superclass A's constructor is invoked first. It invokes setI(20). The setI method in B is used because object created is new B(). The setI method in B assigns 3 * 20 to i. So it displays i from A is 60.

Section 11.6 The Object Class and Its toString() Method

11.15 Analyze the following code:

```java
public class Test {
  public static void main(String[] args) {
    Object a1 = new A();
    Object a2 = new Object();
    System.out.println(a1);
    System.out.println(a2);
  }
}

class A {
  int x;

  @Override
  public String toString() {
    return "A's x is " + x;
  }
}
```

A. The program cannot be compiled, because System.out.println(a1) is wrong and it should be replaced by System.out.println(a1.toString());

B. When executing System.out.println(a1), the toString() method in the Object class is invoked.

[✓ C. When executing System.out.println(a2), the toString() method in the Object class is invoked.]

[✓ D. When executing System.out.println(a1), the toString() method in the A class is invoked.]

Your answer is correct

Explanation: Since a1 is an instance of A, the toString() method in the A class is invoked at runtime.

Section 11.7 Polymorphism

11.16 Given the following code, find the compile error.

```java
public class Test {
  public static void main(String[] args) {
    m(new GraduateStudent());
    m(new Student());
    m(new Person());
    m(new Object());
  }

  public static void m(Student x) {
    System.out.println(x.toString());
  }
}

class GraduateStudent extends Student {
}

class Student extends Person {
  @Override
  public String toString() {
    return "Student";
  }
}

class Person extends Object {
  @Override
  public String toString() {
    return "Person";
  }
}
```

- A. m(new GraduateStudent()) causes an error
- B. m(new Student()) causes an error
- [✓ C. m(new Person()) causes an error]
- [✓ D. m(new Object()) causes an error]

Your answer is correct
Explanation: You cannot pass a supertype variable to a subtype without explicit casting.

Section 11.8 Dynamic Binding

11.17 What is the output of the following code?

```
public class Test {
  public static void main(String[] args) {
    new Person().printPerson();
    new Student().printPerson();
  }
}

class Student extends Person {
  @Override
  public String getInfo() {
    return "Student";
  }
}

class Person {
  public String getInfo() {
    return "Person";
  }

  public void printPerson() {
    System.out.println(getInfo());
  }
}
```

- A. Person Person
- [✓ B. Person Student]
- C. Student Student
- D. Student Person

Your answer is correct

11.18 What is the output of the following code?

```
public class Test {
  public static void main(String[] args) {
    new Person().printPerson();
    new Student().printPerson();
  }
}

class Student extends Person {
  private String getInfo() {
    return "Student";
  }
}

class Person {
  private String getInfo() {
    return "Person";
  }

  public void printPerson() {
    System.out.println(getInfo());
  }
}
```

[✓ A. Person Person]
B. Person Student
C. Student Student
D. Student Person

Your answer is correct ✓
Explanation: Note that the getInfo method is private in Person. It is not known to the outside of the class. This is the method invoked from the printPerson() method.

11.19 Which of the following statements is false?

A. You can always pass an instance of a subclass to a parameter of its superclass type. This feature is known as polymorphism.
B. The compiler finds a matching method according to parameter type, number of parameters, and order of the parameters at compile time.
C. A method may be implemented in several subclasses. The Java Virtual Machine dynamically binds the implementation of the method at runtime.
[✓ D. Dynamic binding can apply to static methods.]
E. Dynamic binding can apply to instance methods.

Your answer is correct ✓
Explanation: Dynamic binding is applied to instance methods, not static methods. Static methods are bound in the compile time.

Section 11.9 Casting Objects and the instanceof Operator

11.20 Which of the following are Java keywords?

A. instanceOf
[✓ B. instanceof]
C. cast
D. casting

Your answer is correct ✓
Explanation: A simple rule: the keywords are all in lowercase.

11.21 Assume Cylinder is a subtype of Circle. Analyze the following code:

```
Cylinder cy = new Cylinder(1, 1);
Circle c = cy;
```

A. The code has a compile error.
B. The code has a runtime error.
[✓ C. The code is fine.]

Your answer is correct ✓
Explanation: You can assign a variable of a subtype to a supertype.

11.22 Assume Cylinder is a subtype of Circle. Analyze the following code:

```
Circle c = new Circle (5);
Cylinder c = cy;
```

[✓ A. The code has a compile error.]
B. The code has a runtime error.
C. The code is fine.

Your answer is correct ✓
Explanation: You cannot assign a variable of a supertype to a subtype without explicit casting.

11.23 Given the following classes and their objects:

```
class C1 {};
class C2 extends C1 {};
class C3 extends C1 {};

C2 c2 = new C2();
C3 c3 = new C3();

Analyze the following statement:
```

```
c2 = (C2)((C1)c3);
```

A. c3 is cast into c2 successfully.
[✓ B.] You will get a runtime error because you cannot cast objects from sibling classes.
C. You will get a runtime error because the Java runtime system cannot perform multiple casting in nested form.
D. The statement is correct.

Your answer is correct ✓

11.24 Given the following code, which of the following expressions evaluates to false?

```
class C1 {}
class C2 extends C1 { }
class C3 extends C2 { }
class C4 extends C1 {}

C1 c1 = new C1();
C2 c2 = new C2();
C3 c3 = new C3();
C4 c4 = new C4();
```

A. c1 instanceof C1
B. c2 instanceof C1
C. c3 instanceof C1
[✓ D.] c4 instanceof C2

Your answer is correct ✓

11.25 Analyze the following code:

```
public class Test {
  public static void main(String[] args) {
    String s = new String("Welcome to Java");
    Object o = s;
    String d = (String)o;
  }
}
```

A. When assigning s to o in Object o = s, a new object is created.
B. When casting o to s in String d = (String)o, a new object is created.
C. When casting o to s in String d = (String)o, the contents of o is changed.
[✓ D.] s, o, and d reference the same String object.

Your answer is correct ✓
Explanation: Casting object reference variable does not affect the contents of the object.

11.26 You can assign _________ to a variable of Object[] type.

A. new char[100]
B. new int[100]
C. new double[100]
[✓ D.] new String[100]
[✓ E.] new java.util.Date[100]

Your answer is correct ✓
Explanation: Primitive data type array is not compatible with Object[].

Section 11.10 The Object?s equals() Method

11.27 The equals method is defined in the Object class. Which of the following is correct to override it in the String class?

A. public boolean equals(String other)
[✓ B.] public boolean equals(Object other)
C. public static boolean equals(String other)
D. public static boolean equals(Object other)

Your answer is correct ✓

11.28 Which of the following statements are true?

[✓ A.] Override the equals(Object) method in the Object class whenever possible.
[✓ B.] Override the toString() method in the Object class whenever possible.
[✓ C.] A public default no-arg constructor is assumed if no constructors are defined explicitly.

✓ D. You should follow standard Java programming style and naming conventions. Choose informative names for classes, data fields, and methods.

Your answer is correct ✓

11.29 What is the output of the following code?

```
public class Test {
  public static void main(String[] args) {
    Object o1 = new Object();
    Object o2 = new Object();
    System.out.print((o1 == o2) + " " + (o1.equals(o2)));
  }
}
```

● A. false false
○ B. true true
○ C. false true
○ D. true false

Your answer is correct ✓
Explanation: o1 == o2 is false, since o1 and o2 are two different objects. o1.equals(o2) is false since the equals method returns o1 == o2 in the Object class.

11.30 What is the output of the following code?

```
public class Test {
  public static void main(String[] args) {
    String s1 = new String("Java");
    String s2 = new String("Java");
    System.out.print((s1 == s2) + " " + (s1.equals(s2)));
  }
}
```

○ A. false false
○ B. true true
● C. false true
○ D. true false

Your answer is correct ✓
Explanation: s1 == s2 is false, since s1 and s2 are two different objects. s1.equals(s2) is true since the equals method returns true if two strings have the same content.

11.31 Given two reference variables t1 and t2, if t1 == t2 is true, t1.equals(t2) must be ___________.

● A. true
○ B. false

Your answer is correct ✓

11.32 Given two reference variables t1 and t2, if t1.equals(t2) is true, t1 == t2 ___________.

○ A. is always true
○ B. is always false
● C. may be true or false

Your answer is correct ✓
Explanation: Two different objects may be equal with the same contents.

11.33 Analyze the following code.

```
// Program 1:
public class Test {
  public static void main(String[] args) {
    Object a1 = new A();
    Object a2 = new A();
    System.out.println(a1.equals(a2));
  }
}

class A {
  int x;

  public boolean equals(Object a) {
    return this.x == ((A)a).x;
  }
}
```

```
// Program 2:
public class Test {
  public static void main(String[] args) {
    Object a1 = new A();
    Object a2 = new A();
    System.out.println(a1.equals(a2));
  }
}

class A {
  int x;

  public boolean equals(A a) {
    return this.x == a.x;
  }
}
```

- A. Program 1 displays true and Program 2 displays true
- B. Program 1 displays false and Program 2 displays true
- [✓ C. Program 1 displays true and Program 2 displays false]
- D. Program 1 displays false and Program 2 displays false

Your answer is correct ✓
Explanation: In Program 1, the equals method in the Object class is overridden. a1.equals(a2) invokes this method. It returns true. In Program 2, the equals method in the Object class is not overridden. a1.equals(a2) invokes the equals method defined in the Object class, which returns false in this case.

11.34 Analyze the following code.

```
// Program 1:
public class Test {
  public static void main(String[] args) {
    Object a1 = new A();
    Object a2 = new A();
    System.out.println(a1.equals(a2));
  }
}

class A {
  int x;

  public boolean equals(A a) {
    return this.x == a.x;
  }
}


// Program 2:
public class Test {
  public static void main(String[] args) {
    A a1 = new A();
    A a2 = new A();
    System.out.println(a1.equals(a2));
  }
}

class A {
  int x;

  public boolean equals(A a) {
    return this.x == a.x;
  }
}
```

- A. Program 1 displays true and Program 2 displays true
- [✓ B. Program 1 displays false and Program 2 displays true]
- C. Program 1 displays true and Program 2 displays false
- D. Program 1 displays false and Program 2 displays false

Your answer is correct ✓
Explanation: In Program 1, the equals method in the Object class is invoked. In Program 2, the equals method in the class A is invoked. There are now two overloaded methods available in the class A. i.e. public boolean equals(Object a) and public boolean equals(A a). Which of the two is used by a1.equals(a2) is determined at compile time. a1.equals(a2) in Program 1 matches the equals method defined in Object and a1.equals(a2) in Program 2 matches the equals method defined in the class A.

11.35 Analyze the following code.

```
// Program 1
public class Test {
  public static void main(String[] args) {
    Object a1 = new A();
    Object a2 = new A();
    System.out.println(((A)a1).equals((A)a2));
  }
}

class A {
  int x;

  public boolean equals(A a) {
    return this.x == a.x;
  }
}


// Program 2
public class Test {
  public static void main(String[] args) {
    A a1 = new A();
    A a2 = new A();
    System.out.println(a1.equals(a2));
  }
}

class A {
  int x;

  public boolean equals(A a) {
    return this.x == a.x;
  }
}
```

[✓ A. Program 1 displays true and Program 2 displays true]
B. Program 1 displays false and Program 2 displays true
C. Program 1 displays true and Program 2 displays false
D. Program 1 displays false and Program 2 displays false

Your answer is correct ✓
Explanation: In Program 1, ((A)a1).equals((A)a2) matches the equals(A a) method in the class A.

Section 11.11 The ArrayList Class

11.36 You can create an ArrayList using _________.

A. new ArrayList[]
B. new ArrayList[100]
[✓ C. new ArrayList<>()]
D. ArrayList()

Your answer is correct ✓

11.37 Invoking _________ removes all elements in an ArrayList x.

A. x.remove()
B. x.clean()
C. x.delete()
D. x.empty()
[✓ E. x.clear()]

Your answer is correct ✓

11.38 Suppose ArrayList x contains two strings [Beijing, Singapore]. Which of the following methods will cause the list to become [Beijing, Chicago, Singapore]?

A. x.add("Chicago")
B. x.add(0, "Chicago")
[✓ C. x.add(1, "Chicago")]
D. x.add(2, "Chicago")

Your answer is correct ✓

11.39 Suppose ArrayList x contains two strings [Beijing, Singapore]. Which of the following method will cause the list to become [Beijing]?

[✓ A. x.remove("Singapore")]
B. x.remove(0)
[✓ C. x.remove(1)]
D. x.remove(2)

Your answer is correct ✓

11.40 Suppose ArrayList x contains two strings [Beijing, Singapore]. Which of the following method will cause runtime errors?

☐ A. x.get(1)
☑ B. x.set(2, "New York");
☑ C. x.get(2)
☑ D. x.remove(2)
☐ E. x.size()

Your answer is correct ✓
Explanation: There is no element at index 2.

11.41 Invoking _________ returns the first element in an ArrayList x.

○ A. x.first()
◉ B. x.get(0)
○ C. x.get(1)
○ D. x.get()

Your answer is correct ✓

11.42 Invoking _________ returns the number of the elements in an ArrayList x.

○ A. x.getSize()
○ B. x.getLength(0)
○ C. x.length(1)
◉ D. x.size()

Your answer is correct ✓

11.43 Analyze the following code:

```
ArrayList<String> list = new ArrayList<String>();
list.add("Beijing");
list.add("Tokyo");
list.add("Shanghai");
list.set(3, "Hong Kong");
```

☑ A. The last line in the code causes a runtime error because there is no element at index 3 in the array list.
☐ B. The last line in the code has a compile error because there is no element at index 3 in the array list.
☑ C. If you replace the last line by list.add(3, "Hong Kong"), the code will compile and run fine.
☐ D. If you replace the last line by list.add(4, "Hong Kong"), the code will compile and run fine.

Your answer is correct ✓
Explanation: There is no element at index 3.

11.44 What is the output of the following code?

```
ArrayList<java.util.Date> list = new ArrayList<java.util.Date>();
java.util.Date d = new java.util.Date();
list.add(d);
list.add(d);
System.out.println((list.get(0) == list.get(1)) + " "
  + (list.get(0)).equals(list.get(1)));
```

○ A. true false
○ B. false true
◉ C. true true
○ D. false false

Your answer is correct ✓
Explanation: list.get(0) and list.get(1) point to the same object.

11.45 What is the output of the following code?

```
ArrayList<String> list = new ArrayList<String>();
String s1 = new String("Java");
String s2 = new String("Java");
list.add(s1);
list.add(s2);
```

```
System.out.println((list.get(0) == list.get(1)) + " "
  + (list.get(0)).equals(list.get(1)));
```

A. true false

[✓ B. false true]

C. true true

D. false false

Your answer is correct ✓
Explanation: list.get(0) and list.get(1) point to two different objects with the same string contents.

11.46 Suppose an ArrayList list contains {"red", "green", "red", "green"}. What is the list after the following code?

```
list.remove("red");
```

A. {"red", "green", "red", "green"}

[✓ B. {"green", "red", "green"}]

C. {"green", "green"}

D. {"red", "green", "green"}

Your answer is correct ✓

11.47 Suppose an ArrayList list contains {"red", "red", "green"}. What is the list after the following code?

```
String element = "red";
for (int i = 0; i < list.size(); i++)
  if (list.get(i).equals(element))
    list.remove(element);
```

A. {"red", "red", "green"}

[✓ B. {"red", "green"}]

C. {"green"}

D. {}

Your answer is correct ✓

11.48 Suppose an ArrayList list contains {"red", "red", "green"}. What is the list after the following code?

```
String element = "red";
for (int i = 0; i < list.size(); i++)
  if (list.get(i).equals(element)) {
    list.remove(element);
    i--;
  }
```

A. {"red", "red", "green"}

B. {"red", "green"}

[✓ C. {"green"}]

D. {}

Your answer is correct ✓

11.49 Suppose an ArrayList list contains {"red", "red", "green"}. What is the list after the following code?

```
String element = "red";
for (int i = list.size() - 1; i >= 0; i--)
  if (list.get(i).equals(element))
    list.remove(element);
```

A. {"red", "red", "green"}

B. {"red", "green"}

[✓ C. {"green"}]

D. {}

Your answer is correct ✓

11.50 The output from the following code is __________.

```
java.util.ArrayList<String> list
  = new java.util.ArrayList<String>();
list.add("New York");
java.util.ArrayList<String> list1 = list;
list.add("Atlanta");
list1.add("Dallas");
System.out.println(list1);
```

A. [New York]

B. [New York, Atlanta]

[✓ C. [New York, Atlanta, Dallas]]

D. [New York, Dallas]

Your answer is correct

Section 11.12 Useful Methods for Lists

11.51 Show the output of the following code:

```java
String[] array = {"red", "green", "blue"};
ArrayList<String> list = new ArrayList<>(Arrays.asList(array));
list.add(0, "red");
System.out.println(list);
```

A. ["red", "green", "blue", "red"]

B. ["red", "green", "blue"]

[✓ C. ["red", "red", "green", "blue"]]

D. ["red", "green", "red", "blue"]

Your answer is correct

11.52 Analyze the following code:

```java
Double[] array = {1, 2, 3};
ArrayList<Double> list = new ArrayList<>(Arrays.asList(array));
System.out.println(list);
```

A. The code is correct and displays [1, 2, 3].

B. The code is correct and displays [1.0, 2.0, 3.0].

[✓ C. The code has a compile error because an integer such as 1 is automatically converted into an Integer object, but the array element type is Double.]

D. The code has a compile error because asList(array) requires that the array elements are objects.

Your answer is correct

11.53 Analyze the following code:

```java
double[] array = {1, 2, 3};
ArrayList<Double> list = new ArrayList<>(Arrays.asList(array));
System.out.println(list);
```

A. The code is correct and displays [1, 2, 3].

B. The code is correct and displays [1.0, 2.0, 3.0].

C. The code has a compile error because an integer such as 1 is automatically converted into an Integer object, but the array element type is Double.

[✓ D. The code has a compile error because asList(array) requires that the array elements are objects.]

Your answer is correct

11.54 Analyze the following code:

```java
double[] c = {1, 2, 3};
System.out.println(java.util.Collections.max(c));
```

A. The code is correct and displays 3.

B. The code is correct and displays 3.0.

[✓ C. The code has a compile error on Collections.max(c). c cannot be an array.]

D. The code has a compile error on double[] c = {1, 2, 3}.

Your answer is correct

11.55 Analyze the following code:

```java
Integer[] c = {3, 5};
java.util.Collections.shuffle(c);
System.out.println(java.util.Arrays.toString(c));
```

A. The code is correct and displays [3, 5].

B. The code is correct and displays [5, 3].

[✓ C. The code has a compile error on Collections.shuffle(c). c cannot be an array.]

D. The code has a compile error on Integer[] c = {3, 5}.

Your answer is correct

Section 11.14 The protected Data and Methods

11.56 What modifier should you use on a class so that a class in the same package can access it but a class (including a subclass) in a different package cannot access it?

- A. public
- B. private
- C. protected
- [✓ D. Use the default modifier.]

Your answer is correct ✓

11.57 What modifier should you use on the members of a class so that they are not accessible to another class in a different package, but are accessible to any subclasses in any package?

- A. public
- B. private
- [✓ C. protected]
- D. Use the default modifier.

Your answer is correct ✓

11.58 The visibility of these modifiers increases in this order:

- A. private, protected, none (if no modifier is used), and public.
- [✓ B. private, none (if no modifier is used), protected, and public.]
- C. none (if no modifier is used), private, protected, and public.
- D. none (if no modifier is used), protected, private, and public.

Your answer is correct ✓

11.59 A class design requires that a particular member variable must be accessible by any subclasses of this class, but otherwise not by classes which are not members of the same package. What should be done to achieve this?

- A. The variable should be marked public.
- B. The variable should be marked private.
- [✓ C. The variable should be marked protected.]
- D. The variable should have no special access modifier.
- E. The variable should be marked private and an accessor method provided.

Your answer is correct ✓
Explanation: See the section on the protected modifier.

11.60 Which of the following statements is false?

- A. A public class can be accessed by a class from a different package.
- B. A private method cannot be accessed by a class in a different package.
- C. A protected method can be accessed by a subclass in a different package.
- [✓ D. A method with no visibility modifier can be accessed by a class in a different package.]

Your answer is correct ✓

11.61 Which statements are most accurate regarding the following classes?

```java
class A {
  private int i;
  protected int j;
}

class B extends A {
  private int k;
  protected int m;
}
```

- [✓ A. An object of B contains data fields i, j, k, m.]
- B. An object of B contains data fields j, k, m.
- C. An object of B contains data fields j, m.
- D. An object of B contains data fields k, m.

Your answer is correct ✓
Explanation: The data fields in a superclass are contained in a subclass. Whether the data fields in a superclass can be accessed in a subclass is a visibility issue. A private data field in a superclass cannot be directly accessed in a subclass, but the data field may have the getter or setter methods, which can be used to get or set a data field value.

11.62 Which statements are most accurate regarding the following classes?

```
class A {
  private int i;
  protected int j;
}

class B extends A {
  private int k;
  protected int m;

  // some methods omitted
}
```

- ○ A. In the class B, an instance method can only access i, j, k, m.
- [✓ B. In the class B, an instance method can only access j, k, m.]
- ○ C. In the class B, an instance method can only access j, m.
- ○ D. In the class B, an instance method can only access k, m.

Your answer is correct ✓

Section 11.15 Preventing Extending and Overriding

11.63 Which of the following classes cannot be extended?

- ○ A. class A { }
- ○ B. class A { private A() { }}
- [✓ C. final class A { }]
- ○ D. class A { protected A() { }}

Your answer is correct ✓