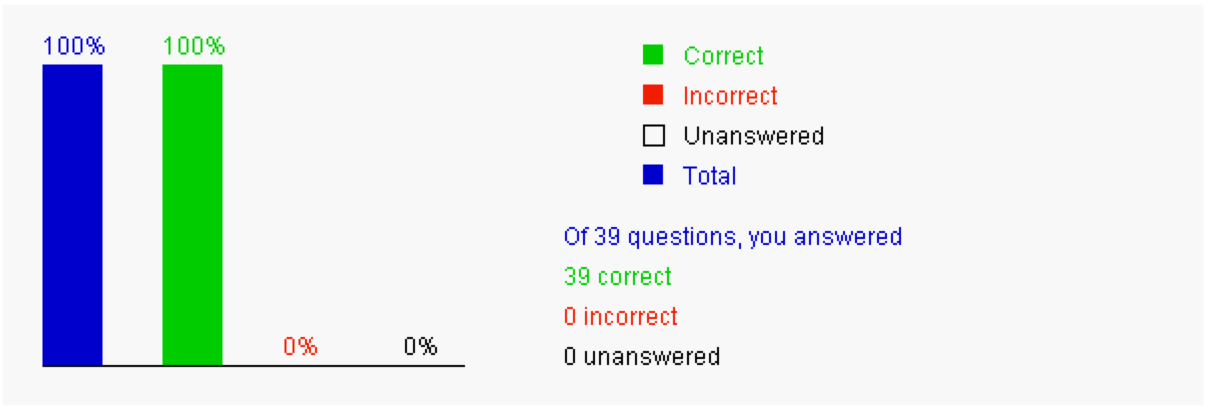
Introduction to Java Programming, Includes Data Structures, Eleventh Edition, Y. Daniel Liang

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 13 Abstract Classes and Interfaces



Of 39 questions, you answered
39 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 13.2 Abstract Classes

13.1 Which of the following class definitions defines a legal abstract class?

- A. class A { abstract void unfinished() { } }
- B. class A { abstract void unfinished(); }
- [✓ C. abstract class A { abstract void unfinished(); }]
- D. public class abstract A { abstract void unfinished(); }

Your answer is correct ✓
Explanation: In A and B, abstract keyword is missing for the class. In D, class and abstract are in a wrong order. The correct answer is C.

13.2 Which of the following declares an abstract method in an abstract Java class?

- A. public abstract method();
- [✓ B. public abstract void method();]
- C. public void abstract method();
- D. public void method() {}
- E. public abstract void method() {}

Your answer is correct ✓
Explanation: In A, the method has no return type. In C, void and abstract are in a wrong order. In D, the abstract keyword is missing. In E, the body {} should be removed. The correct answer is B.

13.3 Which of the following statements regarding abstract methods is false?

- [✓ A. An abstract class can have instances created using the constructor of the abstract class.]
- B. An abstract class can be extended.
- C. A subclass of a non-abstract superclass can be abstract.
- D. A subclass can override a concrete method in a superclass to declare it abstract.
- E. An abstract class can be used as a data type.

Your answer is correct ✓
Explanation: A is wrong because you cannot create an instance using the constructor from an abstract class.

13.4 Which of the following statements regarding abstract methods is false?

- A. Abstract classes have constructors.
- B. A class that contains abstract methods must be abstract.
- C. It is possible to declare an abstract class that contains no abstract methods.
- D. An abstract method cannot be contained in a nonabstract class.
- [✓ E. A data field can be declared abstract.]

Your answer is correct ✓
Explanation: E is wrong, because a data field cannot be declared abstract. Only methods and classes can be declared abstract.

13.5 Suppose A is an abstract class, B is a concrete subclass of A, and both A and B have a no-arg constructor. Which of the following is correct?

- A. A a = new A();

[✓ B.] A a = new B();

C. B b = new A();

[✓ D.] B b = new B();

Your answer is correct

Explanation: Since B is a concrete class with a no-arg constructor, d is correct. Since an instance of B is also an instance of A, b is also correct.

13.6 What is the output of running class Test?

```
public class Test {
  public static void main(String[] args) {
    new Circle9();
  }
}

public abstract class GeometricObject {
  protected GeometricObject() {
    System.out.print("A");
  }

  protected GeometricObject(String color, boolean filled) {
    System.out.print("B");
  }
}

public class Circle9 extends GeometricObject {
  /** No-arg constructor */
  public Circle9() {
    this(1.0);
    System.out.print("C");
  }

  /** Construct circle with a specified radius */
  public Circle9(double radius) {
    this(radius, "white", false);
    System.out.print("D");
  }

  /** Construct a circle with specified radius, filled, and color */
  public Circle9(double radius, String color, boolean filled) {
    super(color, filled);
    System.out.print("E");
  }
}
```

A. ABCD

B. BACD

C. CBAE

D. AEDC

[✓ E.] BEDC

Your answer is correct

Section 13.3 Case Study: the Abstract Number Class

13.7 The java.lang.Number and its subclasses are introduced in Chapter 11. Analyze the following code.

```
Number numberRef = new Integer(0);
Double doubleRef = (Double)numberRef;
```

A. There is no such class named Integer. You should use the class Int.

B. The compiler detects that numberRef is not an instance of Double.

[✓ C.] A runtime class casting exception occurs, since numberRef is not an instance of Double.

D. The program runs fine, since Integer is a subclass of Double.

E. You can convert an int to double, so you can cast an Integer instance to a Double instance.

Your answer is correct

13.8 Analyze the following code.

```
Number[] numberArray = new Integer[2];
numberArray[0] = new Double(1.5);
```

A. You cannot use Number as a data type since it is an abstract class.

B. Since each element of numberArray is of the Number type, you cannot assign an Integer object to it.

C. Since each element of numberArray is of the Number type, you cannot assign a Double object to it.

[✓ D.] At runtime, new Integer[2] is assigned to numberArray. This makes each element of numberArray an Integer object. So you cannot assign a Double object to it.

Your answer is correct

13.9 Analyze the following code. Which of the following statements is correct?

```
public class Test {
  public static void main(String[] args) {
    Number x = new Integer(3);
    System.out.println(x.intValue());
    System.out.println(x.compareTo(new Integer(4)));
  }
}
```

- A. The program has a compile error because an Integer instance cannot be assigned to a Number variable.
- B. The program has a compile error because intValue is an abstract method in Number.
- [✓ C. The program has a compile error because x does not have the compareTo method.]
- D. The program compiles and runs fine.

Your answer is correct

13.10 Analyze the following code. Which of the following statements is correct?

```
public class Test {
  public static void main(String[] args) {
    Number x = new Integer(3);
    System.out.println(x.intValue());
    System.out.println((Integer)x.compareTo(new Integer(4)));
  }
}
```

- A. The program has a compile error because an Integer instance cannot be assigned to a Number variable.
- B. The program has a compile error because intValue is an abstract method in Number.
- C. The program has a compile error because x cannot be cast into Integer.
- [✓ D. The program has a compile error because the member access operator (.) is executed before the casting operator.]
- E. The program compiles and runs fine.

Your answer is correct

13.11 Which of the following statements is incorrect?

- [✓ A. Integer i = 4.5;]
- B. Double i = 4.5;
- C. Object i = 4.5;
- D. Number i = 4.5;

Your answer is correct

Explanation: i is an Integer. You cannot assign 4.5 to i. A is incorrect. B, C, and D are correct. 4.5 is autoboxed to new Double(4.5). A Double object is an instance of Object and Double.

Section 13.4 Case Study: Calendar and GregorianCalendar

13.12 The java.util.Calendar and java.util.GregorianCalendar classes are introduced in Chapter 11. Analyze the following code. Which of the following statements is correct?

```
1. import java.util.*;
2. public class Test {
3.   public static void main(String[] args) {
4.     Calendar[] calendars = new Calendar[10];
5.     calendars[0] = new Calendar();
6.     calendars[1] = new GregorianCalendar();
7.   }
8. }
```

- A. The program has a compile error on Line 4 because java.util.Calendar is an abstract class.
- [✓ B. The program has a compile error on Line 5 because java.util.Calendar is an abstract class.]
- C. The program has a compile error on Line 6 because Calendar[1] is not of a GregorianCalendar type.
- D. The program has no compile errors.

Your answer is correct

Explanation: (A) is incorrect since it is OK to use abstract class as data type for arrays. new Calendar[10] does not create Calendar objects. It just creates an array with 10 elements, each of which can reference to a Calendar object. (B) is correct since you cannot create an object from an abstract class. (C) is incorrect since it is fine to create a GregorianCalendar object and assign its reference to a variable of its superclass type.

13.13 Assume Calendar calendar = new GregorianCalendar(). __________ returns the month of the year.

- [✓ A. calendar.get(Calendar.MONTH)]
- B. calendar.get(Calendar.MONTH_OF_YEAR)

- ○ C. calendar.get(Calendar.WEEK_OF_MONTH)
- ○ D. calendar.get(Calendar.WEEK_OF_YEAR)

Your answer is correct ✓
Explanation: B is wrong since Calendar.MONTH_OF_YEAR is not a valid field.

13.14 Assume Calendar calendar = new GregorianCalendar(). __________ returns the week of the year.

- ○ A. calendar.get(Calendar.MONTH)
- ○ B. calendar.get(Calendar.MONTH_OF_YEAR)
- ○ C. calendar.get(Calendar.WEEK_OF_MONTH)
- ● D. calendar.get(Calendar.WEEK_OF_YEAR)

Your answer is correct ✓

13.15 Assume Calendar calendar = new GregorianCalendar(). __________ returns the number of days in a month.

- ○ A. calendar.get(Calendar.MONTH)
- ○ B. calendar.get(Calendar.MONTH_OF_YEAR)
- ○ C. calendar.get(Calendar.WEEK_OF_MONTH)
- ○ D. calendar.get(Calendar.WEEK_OF_YEAR)
- ● E. calendar.getActualMaximum(Calendar.DAY_OF_MONTH)

Your answer is correct ✓

Section 13.5 Interfaces

13.16 Which of the following is a correct interface?

- ○ A. interface A { void print() { }; }
- ○ B. abstract interface A { print(); }
- ○ C. abstract interface A { abstract void print() { };}
- ● D. interface A { void print();}

Your answer is correct ✓
Explanation: In A, the print() method in the interface is a concrete method. In B and C, the abstract keyword is used before the interface, which is wrong. The correct answer is D.

13.17 Which of the following are incorrect?

- ☐ A. An abstract class contains constructors.
- ☐ B. The constructors in an abstract class should be protected.
- ☑ C. The constructors in an abstract class are private.
- ☑ D. You may declare a final abstract class.
- ☑ E. An interface may contain constructors.

Your answer is correct ✓
Explanation: A and B are correct. C is wrong. D is wrong. E is wrong because an interface does not have constructors.

13.18 _______ is not a reference type.

- ○ A. A class type
- ○ B. An interface type
- ○ C. An array type
- ● D. A primitive type

Your answer is correct ✓
Explanation: Objects and arrays are reference types. Primitive types are not.

13.19 Show the output of running the class Test in the following code lines:

```java
interface A {
}

class C {
}

class B extends D implements A {
}

public class Test {
  public static void main(String[] args) {
    B b = new B();
    if (b instanceof A)
      System.out.println("b is an instance of A");
    if (b instanceof C)
      System.out.println("b is an instance of C");
  }
}

class D extends C {
}
```

A. Nothing.
B. b is an instance of A.
C. b is an instance of C.
[✓ D. b is an instance of A followed by b is an instance of C.]

Your answer is correct

13.20 Suppose A is an interface, B is a concrete class with a no-arg constructor that implements A. Which of the following is correct?

A. A a = new A();
[✓ B. A a = new B();]
C. B b = new A();
[✓ D. B b = new B();]

Your answer is correct
Explanation: Since B is a concrete class with a no-arg constructor, d is correct. Since an instance of B is also an instance of A, b is also correct.

Section 13.6 The Comparable Interface

13.21 Analyze the following code:

```
public class Test1  {
  public Object max(Object o1, Object o2) {
    if ((Comparable)o1.compareTo(o2) >= 0) {
      return o1;
    }
    else {
      return o2;
    }
  }
}
```

A. The program has a compile error because Test1 does not have a main method.
[✓ B. The program has a compile error because o1 is an Object instance and it does not have the compareTo method.]
C. The program has a compile error because you cannot cast an Object instance o1 into Comparable.
[✓ D. The program would compile if ((Comparable)o1.compareTo(o2) >= 0) is replaced by (((Comparable)o1).compareTo(o2) >= 0).]

Your answer is correct
Explanation: The . operator is performed before casting.

13.22 Which of the following statements are true?

[✓ A. The String class implements Comparable.]
[✓ B. The Date class implements Comparable.]
[✓ C. The Double class implements Comparable.]
[✓ D. The BigInteger class implements Comparable.]

Your answer is correct
Explanation: Many classes in the Java API implements the Comparable interface if the objects of the class can be compared. Strings, Dates, Doubles, and BigIntegers can all be compared. These classes implement the Comparable interface.

13.23 Analyze the following code.

```
1. public class Test  {
2.   public static void main(String[] args) {
3.     Fruit[] fruits = {new Fruit(2), new Fruit(3), new Fruit(1)};
4.     java.util.Arrays.sort(fruits);
5.   }
6. }

class Fruit {
  private double weight;

  public Fruit(double weight) {
      this.weight = weight;
    }
  }
```

- A. The program has a compile error because the Fruit class does not have a no-arg constructor.
- B. The program has a runtime error on Line 3 because the Fruit class does not have a no-arg constructor.
- C. The program has a compile error on Line 4 because the Fruit class does not implement the java.lang.Comparable interface and the Fruit objects are not comparable.
- [✓ D.] The program has a runtime error on Line 4 because the Fruit class does not implement the java.lang.Comparable interface and the Fruit objects are not comparable.

Your answer is correct ✓
Explanation: (A) and (B) are incorrect since it is OK to define a class without a no-arg constructor. (C) is incorrect since it is OK to pass fruits to Arrays.sort(Object[]) without compile errors. (D) is correct because the Arrays.sort method requires the objects in the array to be comparable and their class must implement the java.lang.Comparable interface.

Section 13.7 The Cloneable Interface

13.24 Analyze the following code.

```
public class Test {
  public static void main(String[] args) {
    java.util.Date x = new java.util.Date();
    java.util.Date y = x.clone();
    System.out.println(x = y);
  }
}
```

- A. A java.util.Date object is not cloneable.
- B. x = y in System.out.println(x = y) causes a compile error because you cannot have an assignment statement inside a statement.
- C. x = y in System.out.println(x = y) causes a runtime error because you cannot have an assignment statement inside a statement.
- [✓ D.] The program has a compile error because the return type of the clone() method is java.lang.Object.

Your answer is correct ✓
Explanation: (A) is wrong because Date implements and Cloneable and overrides the clone() method. (B) and (C) are wrong because x = y is an assignment expression, which assigns y to x. (D) is correct. You have to cast it into Date in order to assign it to y.

13.25 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 = (java.util.ArrayList<String>)(list.clone());
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 ✓
Explanation: The code added New York to list and cloned list1 from list. list1 now contains New York. Dallas is added to list1. So list1 contains New Your and Dallas.

13.26 The GeometricObject and Circle classes are defined in this chapter. Analyze the following code. Which statements are correct?

```
public class Test {
  public static void main(String[] args) {
    GeometricObject x = new Circle(3);
    GeometricObject y = (Circle)(x.clone());
    System.out.println(x);
    System.out.println(y);
  }
}
```

- [✓ A.] The program has a compile error because the clone() method is protected in the Object class.
- [✓ B.] After you override the clone() method and make it public in the Circle class, the problem can compile and run just fine, but y is null if Circle does not implement the Cloneable interface.
- [✓ C.] To enable a Circle object to be cloned, the Circle class has to override the clone() method and implement the java.lang.Cloneable interface.
- [✓ D.] If GeometricObject implements Cloneable and Circle overrides the clone() method, the clone() method will work fine to clone Circle objects.

Your answer is correct ✓

Section 13.8 Interfaces vs. Abstract Classes

13.27 Which of the following statements is false?

- A. If you compile an interface without errors, a .class file is created for the interface.

B. If you compile a class without errors but with warnings, a .class file is created.

[✓ C.] If you compile a class with errors, a .class file is created for the class.

D. If you compile an interface without errors, but with warnings, a .class file is created for the interface.

Your answer is correct

Explanation: A .class file is created for each Java class and interface. But if it has a compile error, no .class file is created.

13.28 Which of the following statements are true?

[✓ A.] Inheritance models the is-a relationship between two classes.

[✓ B.] A strong is-a relationship describes a direct inheritance relationship between two classes.

[✓ C.] A weak is-a relationship describes that a class has certain properties.

[✓ D.] A strong is-a relationship can be represented using class inheritance.

[✓ E.] A weak is-a relationship can be represented using interfaces.

Your answer is correct

13.29 What is the best suitable relationship between Employee and Faculty?

A. Composition

B. Aggregation

[✓ C.] Inheritance

D. None.

Your answer is correct

13.30 Assume an employee can work for only one company. What is the best suitable relationship between Company and Employee?

A. None

B. Aggregation

C. Inheritance

[✓ D.] Composition

Your answer is correct

13.31 The relationship between an interface and the class that implements it is

A. Composition

B. Aggregation

[✓ C.] Inheritance

D. None

Your answer is correct

Section 13.9 Case Study: The Rational Class

13.32 The Rational class in this chapter is defined as a subclass of java.lang.Number. Which of the following expressions is correct?

A. Rational.doubleValue();

B. Rational.doubleValue("5/4");

[✓ C.] new Rational(5, 4).doubleValue();

D. new Rational(5, 4).toDoubleValue();

[✓ E.] new Rational(5, 4).intValue();

Your answer is correct

Explanation: A is wrong, because the doubleValue() method is an instance method. B is wrong for two reasons: 1. the doubleValue() method is an instance method; 2. the doubleValue() method does not have arguments. D is wrong becasue there is no method named toDoubleValue().

13.33 The Rational class in this chapter extends java.lang.Number and implements java.lang.Comparable. Analyze the following code.

```java
public class Test {
  public static void main(String[] args) {
    Number[] numbers = {new Rational(1, 2), new Integer(4), new Double(5.6)};
    java.util.Arrays.sort(numbers);
  }
}
```

A. The program has a compile error because numbers is declared as Number[], so you cannot assign {new Rational(1, 2), new Integer(4), new Double(5.6)} to it.

B. The program has a runtime error because numbers is declared as Number[], so you cannot assign {new Rational(1, 2), new Integer(4), new Double(5.6)} to it.

C. The program has a compile error because numbers is declared as Number[], so you cannot pass it to Arrays.sort(Object[]).

[✓ D.] The program has a runtime error because the compareTo methods in Rational, Integer, and Double classes do not compare the value of one type with a value of another type.

Your answer is correct ✓
Explanation: (A) and (B) are incorrect because Rational, Integer, and Double are subclasses of Number and any instances of these classes can be elements of the Number[] array. (C) is incorrect because it is fine to pass an instance of Number[] to a parameter of the Object[] type. (D) is correct because the compareTo method in Rational, Integer, Double only compare two Rational objects, two Integer objects, or two Double objects.

Section 13.10 Class Design Guidelines

13.34 Which of the following statements are true?

- [✓ A.] A class should describe a single entity and all the class operations should logically fit together to support a coherent purpose.
- B. A class should always contain a no-arg constructor.
- C. The constructors must always be public.
- [✓ D.] The constructors may be protected.

Your answer is correct ✓
Explanation: (B) is not true. Most classes have a no-arg constructor. But sometimes, it does not make any sense to provide a no-arg constructor. For example, StringTokenizer does not have a no-arg constructor. (C) is not true. For example, the constructor in the Math class is private to prevent creating instances from the class, because there is no need to create instances for the Math class since all methods are static in Math. The constructors for abstract classes should be protected in most cases.

13.35 Which of the following is poor design?

- [✓ A.] A data field is derived from other data fields in the same class.
- [✓ B.] A method must be invoked after/before invoking another method in the same class.
- [✓ C.] A method is an instance method, but it does not reference any instance data fields or invoke instance methods.
- [✓ D.] A parameter is passed from a constructor to initialize a static data field.

Your answer is correct ✓
Explanation: A is not good because there is no need to define the data field if it can be derived from other data fields. B is not good because it is highly problematic to impose any order for invoking the methods. C is a bad design because in this case the method should be defined as static. D is not good because a static data field is now tied to the creation of a specific object, which should not be the case.

13.36 Polymorphism means ______________.

- A. that data fields should be declared private
- B. that a class can extend another class
- [✓ C.] that a variable of supertype can refer to a subtype object
- D. that a class can contain another class

Your answer is correct ✓

13.37 Encapsulation means ______________.

- [✓ A.] that data fields should be declared private
- B. that a class can extend another class
- C. that a variable of supertype can refer to a subtype object
- D. that a class can contain another class

Your answer is correct ✓

13.38 Inheritance means ______________.

- A. that data fields should be declared private
- [✓ B.] that a class can extend another class
- C. that a variable of supertype can refer to a subtype object
- D. that a class can contain another class

Your answer is correct ✓

13.39 Composition means ______________.

- A. that data fields should be declared private
- B. that a class extends another class
- C. that a variable of supertype refers to a subtype object
- [✓ D.] that a class contains a data field that references another object

Your answer is correct ✓