Due to the print book page limit, we cannot inlcude all good CheckPoint questions in the physical book. The CheckPoint on this Website may contain extra questions not printed in the book. The questions in some sections may have been reordered as a result. Nevertheless, it is easy to find the CheckPoint questions in the book on this Website. Please send suggestions and errata to Dr. Liang at y.daniel.liang@gmail.com. Indicate the book, edition, and question number in your email. Thanks!

Chapter 13 Check Point Questions

Section 13.2

▼13.2.1

Which of the following classes defines a legal abstract class?

```
(a)
class A {
  abstract void unfinished() {
  }
}

(b)
public class abstract A {
  abstract void unfinished();
}

(c)
public class abstract A {
  abstract void unfinished();
}

(d)
abstract class A {
  protected void unfinished();
}

(e)
abstract class A {
  abstract void unfinished();
}

(f)
abstract class A {
  abstract int unfinished();
}
```

e and f

Hide Answer

▼13.2.2

The getArea() and getPerimeter() methods may be removed from the GeometricObject class. What are the benefits of defining getArea() and getPerimeter() as abstract methods in the GeometricObject class?

The benefits are for generic programming. A variable of GeometricObject type can use the getArea and getPerimeter methods at compilation time.

Hide Answer

▼**13.2.3**

True or false?

a. An abstract class can be used just like a nonabstract class except that you cannot use the new operator to create an instance from the abstract class.
b. An abstract class can be extended.
c. A subclass of a nonabstract superclass cannot be abstract.
d. A subclass cannot override a concrete method in a superclass to define it as abstract.
e. An abstract method must be nonstatic.

a. True
b. True
c. False
d. False
e. True

Hide Answer

Section 13.3

▼**13.3.1**

Why do the following two lines of code compile but cause a runtime error?

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

At runtime, JVM attempts to convert numberRef to a Double object, but numberRef is an instance of Integer, not Double.

Hide Answer

▼**13.3.2**

Why do the following two lines of code compile but cause a runtime error?

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

numberArray[0] is of the Integer type. It will be a casting error at runtime.

Hide Answer

▼**13.3.3**

Show the output of the following code.

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

```
3
3.0
```

Hide Answer

▼13.3.4
What is wrong in the following code? (Note that the compareTo method for the Integer and Double classes was introduced in Section 10.7.)

```
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)));
  }
}
```

The program has a syntax error because x does not have the compareTo method.

Hide Answer

▼13.3.5
What is wrong in the following code?

```
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)));
  }
}
```

The program has a syntax error because the member access operator (.) is executed before the casting operator.

Hide Answer

Section 13.4

▼13.4.1
Can you create a Calendar object using the Calendar class?

No. Calendar is an abstract class.

Hide Answer

▼13.4.2
Which method in the Calendar class is abstract?

The add method in the Calendar class is abstract.

Hide Answer

▼13.4.3
How do you create a Calendar object for the current time?

Use the GregorianCalendar class's no-arg constructor, you can create an instance of Calendar.

Hide Answer

▼13.4.4
For a Calendar object c, how do you get its year, month, date, hour, minute, and second?

```
c.get(Calendar.YEAR)
c.get(Calendar.MONTH)
c.get(Calendar.DAY_OF_MONTH)
c.get(Calendar.HOUR)
c.get(Calendar.MINUTE)
c.get(Calendar.SECOND)
```

Hide Answer

Section 13.5

▼13.5.1

Suppose A is an interface. Can you create an instance using new A()?

No

Hide Answer

▼13.5.2

Suppose A is an interface. Can you declare a reference variable x with type A like this?

```
A x;
```

Yes

Hide Answer

▼13.5.3

Which of the following is a correct interface?

```
(a)
interface A {
  void print() { }
}

(b)
abstract interface A {
  abstract void print() { }
}

(c)
abstract interface A {
  print();
}

(d)
interface A {
  void print();
}

(e)
interface A {
  default void print() {
  }
}

(f)
interface A {
  static int get() {
    return 0;
  }
}
```

(a) is wrong, because the print() method has a body.
(b) is wrong, because the interface cannot have the abstract keyword.
(c) is wrong, because the interface cannot have the abstract keyword.
(d) is correct.
(e) is correct. JDK 8 allows default methods in the interface.
(f) is correct. JDK 8 allows static methods in the interface.

Hide Answer

▼13.5.4

Show the error in the following code:

```
interface A {
  void m1();
}

class B implements A {
  void m1() {
    System.out.println("m1");
  }
}
```

All methods defined in an interface are public. When a class implements the interface, the method must be declared public. The visibility cannot be reduced.

Hide Answer

▼13.5.5

The following questions are based on the Edible interface and the classes defined in Listing 13.7 and also assume LittleChicken is a subtype of Chicken. For each question, answer if the code can compile, can run. If not, give a reason. If it runs, give the output.

```
a. Edible x = new Tiger();

b. Edible x = new Chicken();
   System.out.println(x.sound());

c. Edible x = new Chicken();
   System.out.println((Animal)x.sound());

d. Edible x = new Chicken();
   System.out.println(((Animal)x).sound());

e. Edible x = new LittleChicken();
   System.out.println(x.howToEat());

f. LittleChicken x = new Chicken();
```

a. Compile error. Tiger is not Edible.
b. Compile error. x is declared Edible, but Edible does not have the sound() method.
c. Compile error. x.sound() performed first before casting. x does not have the sound() method.
d. Chicken: cock-a-doodle-doo

e. Chicken: Fry it
f. Compile error. Chicken is not a LittleChicken. Cannot assign a Chicken object to a variable of type LittleChicken.

Hide Answer

Section 13.6

▼13.6.1

True or false? If a class implements Comparable, the object of the class can invoke the compareTo method.

True

Hide Answer

▼13.6.2

Which of the following is the correct method header for the compareTo method in the String class?

```
public int compareTo(String o)
public int compareTo(Object o)
```

The first one is correct.

Hide Answer

▼13.6.3

Can the following code be compiled? Why?

```
Integer n1 = new Integer(3);
Object n2 = new Integer(4);
System.out.println(n1.compareTo(n2));
```

n1 is an Integer object whose compareTo method require an Integer argument, but n2 is declared as Object. The compiler will raise an error.

Hide Answer

▼13.6.4

You can define the compareTo method in a class without implementing the Comparable interface. What are the benefits of implementing the Comparable interface?

By implementing the Comparable interface, the object of the class can be passed to a method that requires a Comparable type.

Hide Answer

▼13.6.5

What is wrong in the following code?

```
public class Test {
  public static void main(String[] args) {
    Person[] persons = {new Person(3), new Person(4), new Person(1)};
    java.util.Arrays.sort(persons);
  }
}
```

```
class Person {
  private int id;

  Person(int id) {
    this.id = id;
  }
}
```

A: The Person class does not implement the Comparable interface, two persons can not be compared using the compareTo method.

Hide Answer

▼**13.6.6**
Simplify the code in lines 10-15 in Listing 13.9 using one line of code. Also override the equals method in this class.

```
return getArea() > o.getArea() ? 1 : (getArea() < o.getArea() ? -1 : 0);
```

The equals method can be overridden as follows:

```
@Override
public boolean equals(Object o) {
  return getArea() == ((CompareRectangle)o).getArea();
}
```

Hide Answer

▼**13.6.7**
Listing 13.5 has an error. If you add list.add(new BigInteger("3432323234344343102")); in line 11, you will see that the result is incorrect. This is due to the fact that a double value can have up to 17 significant digits. When invoking doubleValue() on a BigInteger object in line 24, precision is lost. Fix the error by converting the numbers into BigDecimal and compare them using the compareTo method in line 24.

```
      if (new BigDecimal(number + "").compareTo
         (new BigDecimal(list.get(i) + "")) < 0) {
```

The complete code is here:

```
import java.util.ArrayList;
import java.math.*;

public class LargestNumber {
  public static void main(String[] args) {
    ArrayList<Number> list = new ArrayList<>();
    list.add(45); // Add an integer
    list.add(3445.53); // Add a double
    // Add a BigInteger
    list.add(new BigInteger("3432323234344343101"));
    list.add(new BigInteger("3432323234344343102"));

    // Add a BigDecimal
    list.add(new BigDecimal("2.0909090989091343433344343"));

    System.out.println("The largest number is " +
      getLargestNumber(list));
  }

  public static Number getLargestNumber(ArrayList<Number> list) {
    if (list == null || list.size() == 0)
      return null;

    Number number = list.get(0);
    for (int i = 1; i < list.size(); i++)
      if (new BigDecimal(number + "").compareTo
         (new BigDecimal(list.get(i) + "")) < 0) {
        number = list.get(i);
      }

    return number;
  }
}
```

Hide Answer

Section 13.7

▼**13.7.1**
Can a class invoke super.clone() when implementing the clone() method if the class does not implement java.lang.Cloneable? Does the Date class implement Cloneable?

You can invoke super.clone() when implementing the clone() methodif the class does not implement java.lang.Cloneable, as shown below:

```
class Foo {
  int x = 5;
  public Object clone() throws CloneNotSupportedException {
    return super.clone();
  }
}
```

However, it does not actually clone the object. The following code will throw CloneNotSupportedException.

```
public class Test1 {
  public static void main(String[] args) throws CloneNotSupportedException {
    Foo foo = (Foo)new Foo().clone();
    System.out.println(foo.x);
  }
}
```

If the Foo class implements java.lang.Cloneable as follows, the Test1 class will run fine.

```
class Foo implements java.lang.Cloneable {
  int x = 5;
  public Object clone() throws CloneNotSupportedException {
    return super.clone();
  }
}
```

Yes, the Date class implements Cloneable.

Hide Answer

▼13.7.2

What would happen if the House class (defined in Listing 13.11) did not override the clone() method or if House did not implement java.lang.Cloneable?

If the House class does not override the clone() method, the program would receive a syntax error because clone() is protected in java.lang.Object. For example, the following code cannot compile.

```
public class Test1 {
  public static void main(String[] args) throws CloneNotSupportedException {
    House house = (House)new House().clone(); // clone() is not visible
  }
}
```

If House does not implement java.lang.Cloneable, a CloneNotSupportedException would occur when invoking super.clone().

Hide Answer

▼13.7.3

Show the output of the following code:

```
java.util.Date date = new java.util.Date();
java.util.Date date1 = date;
java.util.Date date2 = (java.util.Date)(date.clone());
System.out.println(date == date1);
System.out.println(date == date2);
System.out.println(date.equals(date2));
```

```
            true
            false
            true
```

Hide Answer

▼13.7.4

Show the output of the following code:

```
ArrayList<String> list = new ArrayList<>();
list.add("New York");
ArrayList<String> list1 = list;
ArrayList<String> list2 = (ArrayList<String>)(list.clone());
list.add("Atlanta");
System.out.println(list == list1);
System.out.println(list == list2);
System.out.println("list is " + list);
System.out.println("list1 is " + list1);
System.out.println("list2.get(0) is " + list2.get(0));
System.out.println("list2.size() is " + list2.size());
```

```
true
false
list is [New York, Atlanta]
list1 is [New York, Atlanta]
list2.get(0) is New York
list2.size() is 1
```

Hide Answer

▼13.7.5
What is wrong in the code in (a)? Why does the code in (b) have no compile errors?

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

(b)
public class Test5 {
  public static void main(String[] args) throws CloneNotSupportedException {
    Test5 x = new Test5();
    GeometricObject y = (GeometricObject)x.clone();
  }
}
```

In (a), a compile error is reported because clone() is protected in Object. To enable cloning, do two things: (1) override clone() in the class for the object to be cloned; (2) implement java.lang.Cloneable for the class.

In (b), no compile errors, because Test5 extends Object and the clone() method is defined in the Obejct class, which is visible in Test5. However, when you run the code, a CloneNotSupportedException will be thrown, because the clone() method is not implemented in Test5 and Test5 does not implement the Cloneable interface.

Hide Answer

▼13.7.6
Show the output of the following code.

```
public class Test {
  public static void main(String[] args) {
    House house1 = new House(1, 1750, 50);
    House house2 = (House)house1.clone();
    System.out.println(house1.equals(house2);
  }
}
```

false

Hide Answer

Section 13.8

▼13.8.1
Give an example to show why interfaces are preferred over abstract classes.

See the example in the text in the section on Interfaces vs. Abstract Classes.

Hide Answer

▼13.8.2

Define the terms abstract classes and interfaces. What are the similarities and differences between abstract classes and interfaces?

See the section on Interfaces vs. Abstract Classes.

Hide Answer

▼13.8.3

True or false?
a. An interface is compiled into a separate bytecode file.
b. An interface can have static methods.
c. An interface can extend one or more interfaces.
d. An interface can extend an abstract class.
e. An interface can have default methods.

a. True
b. True in Java 8
c. True
d. False
e. True in Java 8

Hide Answer

Section 13.9

▼13.9.1

Show the output of the following code?

```
Rational r1 = new Rational(-2, 6);
System.out.println(r1.getNumerator());
System.out.println(r1.getDenominator());
System.out.println(r1.intValue());
System.out.println(r1.doubleValue());
```

```
-1
3
0
0.333333333333
```

Hide Answer

▼13.9.2

Why is the following code wrong?

```
Rational r1 = new Rational(-2, 6);
Object r2 = new Rational(1, 45);
System.out.println(r2.compareTo(r1));
```

The Object type r2 does not have the compareTo method.

Hide Answer

▼13.9.3

Why is the following code wrong?

```
Object r1 = new Rational(-2, 6);
```

```
Rational r2 = new Rational(1, 45);
System.out.println(r2.compareTo(r1));
```

The compareTo(Rational) method requires a Rational type object in the parameter in the Rational class.

Hide Answer

▼13.9.4

Simplify the code in lines 82-85 in Listing 13.13 Rational.java using one line of code without using the if statement. Simply the code in lines 110-115 using a conditional operator.

```
public boolean equals(Object o) {
  return (this.subtract((Rational)(other)))
    .getNumerator() == 0;
}

public int compareTo(Rational o) {
  return this.subtract(o).getNumerator() > 0 ? 1 :
    (this.subtract(o).getNumerator() == 0 ? 0 : 1);
}
```

Hide Answer

▼13.9.5

Trace the program carefully and show the output of the following code.

```
Rational r1 = new Rational(1, 2);
Rational r2 = new Rational(1, -2);
System.out.println(r1.add(r2));
```

0/4

Hide Answer

▼13.9.6

The preceding question shows a bug in the toString method. Revise the toString() method to fix the error.

```
public String toString() {
  if (numerator == 0 || denominator == 1)
    return numerator + "";
  else
    return numerator + "/" + denominator;
}
```

Hide Answer

▼13.9.7

What happens if you create a Rational using new Rational(1, 0)? Discuss the appropriate ways for handling this case.

new Rational(1, 0) would create a Rational with denominator 0. The best way to handle this case is to throw an IllegalArgumentException when denominator is 0.

Hide Answer

Section 13.10

▼**13.10.1**

Describe class design guidelines.

See the text.

Hide Answer