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 9 Check Point Questions

Section 9.3

▼9.3.1

Describe the relationship between an object and its defining class.

See the section "Defining Classes for Objects."

Hide Answer

▼9.3.2

How do you define a class?

The syntax to define a class is

```
public class ClassName {
}
```

Hide Answer

▼9.3.3

How do you declare an object's reference variable?

The syntax to declare a reference variable for an object is

```
ClassName v;
```

Hide Answer

▼9.3.4

How do you create an object?

The syntax to create an object is

```
new ClassName();
```

Hide Answer

Section 9.4

▼9.4.1

What are the differences between constructors and methods?

Constructors are special kinds of methods that are called when creating an object using the new operator. Constructors do not have a return type-not even void.

Hide Answer

▼9.4.2

When will a class have a default constructor?

A class has a default constructor only if the class does not define any constructor.

Hide Answer

Section 9.5

▼9.5.1

Which operator is used to access a data field or invoke a method from an object?

The member access operator is used to access a data field or invoke a method from an object.

Hide Answer

▼9.5.2

What is an anonymous object?

An anonymous object is the one that does not have a reference variable referencing it.

Hide Answer

▼9.5.3

What is NullPointerException?

A NullPointerException occurs when a null reference variable is used to access the members of an object.

Hide Answer

▼9.5.4

Is an array an object or a primitive type value? Can an array contain elements of an object type? Describe the default value for the elements of an array.

An array is an object. An array can contain elements of an object type. The default value for the elements of an array is 0 for numeric, false for boolean, '\u0000' for char, null for object element type.

Hide Answer

▼9.5.5

What is wrong with each of the following programs?

(a)

```
1  public class ShowErrors {
2    public static void main(String[] args) {
3      ShowErrors t = new ShowErrors(5);
4    }
5  }
```

(b)

```
1  public class ShowErrors {
2    public static void main(String[] args) {
3      ShowErrors t = new ShowErrors();
4      t.x();
5    }
6  }
```

(c)

```
1  public class ShowErrors {
2    public void method1() {
3      Circle c;
4      System.out.println("What is radius "
5        + c.getRadius());
6      c = new Circle();
7    }
8  }
```

(d)

```
1  public class ShowErrors {
2    public static void main(String[] args) {
3      C c = new C(5.0);
4      System.out.println(c.value);
5    }
6  }
7
8  class C {
9    int value = 2;
10 }
```

(a) There is no such constructor ShowErrors(int) in the ShowErrors class.The ShowErrors class in the book has a default constructor. It is actually same as

```
public class ShowErrors {
  public static void main(String[] args) {
    ShowErrors t = new ShowErrors(5);
  }

  public ShowErrors () {
  }
}
```

On Line 3, new ShowErrors(5) attempts to create an instance using a constructor ShowErrors(int), but the ShowErrors class does not have such a constructor. That is an error.
(b) x() is not a method in the ShowErrors class.The ShowErrors class in the book has a default constructor. It is actually same as

```
public class ShowErrors {
  public static void main(String[] args) {
    ShowErrors t = new ShowErrors();
    t.x();
  }
  public ShowErrors () {
  }
}
```

On Line 4, t.x() is invoked, but the ShowErrors class does not have the method named x(). That is an error.(c) The program compiles fine, but it has a runtime error because variable c is null when the println statement is executed.
(d) new C(5.0) does not match any constructors in class C. The program has a compilation error because class C does not have a constructor with a double argument.

Hide Answer

▼9.5.6

What is wrong in the following code?

```
 1  class Test {
 2    public static void main(String[] args) {
 3      A a = new A();
 4      a.print();
 5    }
 6  }
 7
 8  class A {
 9    String s;
10
11    A(String newS) {
12      s = newS;
13    }
14
15    public void print() {
16      System.out.print(s);
17    }
18  }
```

The program does not compile because new A() is used in class Test, but class A does not have a default constructor. See the second NOTE in the Section, "Constructors."

Hide Answer

▼9.5.7

What is the output of the following code?

```
public class A {
  boolean x;

  public static void main(String[] args) {
    A a = new A();
    System.out.println(a.x);
  }
}
```

false

Hide Answer

Section 9.6

▼9.6.1

How do you create a Date for the current time? How do you display the current time?

Use the Date's no-arg constructor to create a Date for the current time. Use the Date's toString() method to display a string representation for the Date.

Hide Answer

▼9.6.2

How do you create a Point2D? Suppose p1 and p2 are two instances of Point2D, how do you obtain the distance between the two points? How do you obtain the midpoint between the two points?

You can create a Point2D object using its constructor Point2D(x, y) for a point at (x, y). Use p1.distance(p2) to obtain the distance between p1 and p2. Use p1.midPoint(p2) to obtain the mid-point between p1 and p2.

Hide Answer

▼9.6.3

Which packages contain the classes Date, Random, Point2D, System, and Math?

Date is in the java.util package. Random is in the java.util package. Point2D is in the javafx.geometry package. System and Math are in the java.lang package.

Hide Answer

Section 9.7

▼9.7.1

Suppose that the class F is defined in (a). Let f be an instance of F. Which of the statements in (b) are correct?

```
(a)
public class F {
  int i;
  static String s;

  void imethod() {
  }

  static void smethod() {
  }
}

(b)
System.out.println(f.i);
System.out.println(f.s);
f.imethod();
f.smethod();
System.out.println(F.i);
System.out.println(F.s);
F.imethod();
F.smethod();
```

System.out.println(f.i);
Answer: Correct

System.out.println(f.s);
Answer: Correct

f.imethod();
Answer: Correct

f.smethod();
Answer: Correct

System.out.println(F.i);
Answer: Incorrect

System.out.println(F.s);
Answer: Correct

F.imethod();
Answer: Incorrect

```
F.smethod();
Answer: Correct
```

Hide Answer

▼9.7.2

Add the static keyword in the place of ? if appropriate.

```
public class Test {
  int count;

  public ? void main(String[] args) {
    ...
  }

  public ? int getCount() {
    return count;
  }

  public ? int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++)
      result *= i;
    return result;
  }
}
```

Add static in the main method and in the factorial method because these two methods don't need reference any instance objects or invoke any instance methods in the Test class.

Hide Answer

▼9.7.3

1. Can you invoke an instance method or reference an instance variable from a static method?
2. Can you invoke a static method or reference a static variable from an instance method?
3. What is wrong in the following code?

```
 1  public class C {
 2    Circle c = new Circle();
 3
 4    public static void main(String[] args) {
 5      method1();
 6    }
 7
 8    public void method1() {
 9      method2();
10    }
11
12    public static void method2() {
13      System.out.println("What is radius " + c.getRadius());
14    }
15  }
```

1. You cannot invoke an instance method or reference an instance variable from a static method.
2. You can invoke a static method or reference a static variable from an instance method.
3. (a) The main method is static and cannot invoke the instance method method1. (b) c is an instance

variable, which cannot be accessed from the static context in method2.

Hide Answer

Section 9.9

▼9.9.1

What is an accessor method? What is a mutator method? What are the naming conventions for accessor methods and mutator methods?

Accessor method is for retrieving private data value and mutator method is for changing private data value. The naming convention for accessor method is getDataFieldName() for non-boolean values and isDataFieldName() for boolean values. The naming convention for mutator method is setDataFieldName(value).

Hide Answer

▼9.9.2

What are the benefits of data field encapsulation?

Two benefits: (1) for protecting data and (2) for easy to maintain the class.

Hide Answer

▼9.9.3

In the following code, radius is private in the Circle class, and myCircle is an object of the Circle class. Does the highlighted code cause any problems? If so, explain why.

```
public class Circle {
  private double radius = 1;

  /** Find the area of this circle */
  public double getArea() {
    return radius * radius * Math.PI;
  }

  public static void main(String[] args) {
    Circle myCircle = new Circle();
    System.out.println("Radius is " + myCircle.radius);
  }
}
```

Not a problem. Though radius is private, myCircle.radius is used inside the Circle class. Thus, it is fine.

Hide Answer

Section 9.10

▼9.10.1

Describe the difference between passing a parameter of a primitive type and passing a parameter of a reference type. Show the output of the following programs:

```
public class Test {
  public static void main(String[] args) {
    Count myCount = new Count();
    int times = 0;
```

```
    for (int i = 0; i < 100; i++)
      increment(myCount, times);

    System.out.println("count is " + myCount.count);
    System.out.println("times is " + times);
  }

  public static void increment(Count c, int times) {
    c.count++;
    times++;
  }
}

class Count {
  public int count;

  public Count(int c) {
    count = c;
  }

  public Count() {
    count = 1;
  }
}
```

Java uses "pass by value" to pass parameters to a method. When passing a variable of a primitive type to a method, the variable remains unchanged after the method finishes. However, when passing a variable of a reference type to a method, any changes to the object referenced by the variable inside the method are permanent changes to the object referenced by the variable outside of the method. Both the actual parameter and the formal parameter variables reference to the same object.The output of the program is as follows:

```
count 101
times 0
```

Hide Answer

▼9.10.2

Show the output of the following program:

```
public class Test {
  public static void main(String[] args) {
    Circle circle1 = new Circle(1);
    Circle circle2 = new Circle(2);

    swap1(circle1, circle2);
    System.out.println("After swap1: circle1 = " +
    circle1.radius + " circle2 = " + circle2.radius);

    swap2(circle1, circle2);
    System.out.println("After swap2: circle1 = " +
      circle1.radius + " circle2 = " + circle2.radius);
  }

  public static void swap1(Circle x, Circle y) {
    Circle temp = x;
```

```
    x = y;
    y = temp;
  }

  public static void swap2(Circle x, Circle y) {
    double temp = x.radius;
    x.radius = y.radius;
    y.radius = temp;
  }
}

class Circle {
  double radius;

  Circle(double newRadius) {
    radius = newRadius;
  }
}
```

```
After swap1: circle1 = 1.0 circle2 = 2.0
After swap2: circle1 = 2.0 circle2 = 1.0
```

Remark: The reference value of circle1 is passed to x and the reference value of circle2 is passed to y. The contents of the objects are not swapped in the swap1 method. circle1 and circle2 are not swapped. To actually swap the contents of these objects, replace the following three lines

```
Circle temp = x;
x =y;
y =temp;
```

by

```
double temp = x.radius;
x.radius = y.radius;
y.radius = temp;
```

as in swap2.

Hide Answer

▼9.10.3

Show the output of the following code:

```
(a)
public class Test {
  public static void main(String[] args) {
    int[] a = {1, 2};
    swap(a[0], a[1]);
    System.out.println("a[0] = " + a[0]
      + " a[1] = " + a[1]);
  }

  public static void swap(int n1, int n2) {
    int temp = n1;
    n1 = n2;
    n2 = temp;
  }
}
```

```
(b)
public class Test {
  public static void main(String[] args) {
    int[] a = {1, 2};
    swap(a);
    System.out.println("a[0] = " + a[0]
      + " a[1] = " + a[1]);
  }

  public static void swap(int[] a) {
    int temp = a[0];
    a[0] = a[1];
    a[1] = temp;
  }
}

(c)
public class Test {
  public static void main(String[] args) {
    T t = new T();
    swap(t);
    System.out.println("e1 = " + t.e1
       + " e2 = " + t.e2);
  }

  public static void swap(T t) {
    int temp = t.e1;
    t.e1 = t.e2;
    t.e2 = temp;
  }
}

class T {
  int e1 = 1;
  int e2 = 2;
}

(d)
public class Test {
  public static void main(String[] args) {
    T t1 = new T();
    T t2 = new T();
    System.out.println("t1's i = " +
      t1.i + " and j = " + t1.j);
    System.out.println("t2's i = " +
      t2.i + " and j = " + t2.j);
  }
}

class T {
  static int i = 0;
  int j = 0;

  T() {
    i++;
    j = 1;
  }
}
```

a. a[0] = 1 a[1] = 2

```
b. a[0] = 2 a[1] = 1
c. e1 = 2 e2 = 1
d. t1's i = 2 t1's j = 1
   t2's i = 2 t2's j = 1
```

Hide Answer

▼9.10.4

What is the output of the following programs?

```
(a)
import java.util.Date;

public class Test {
  public static void main(String[] args) {
    Date date = null;
    m1(date);
    System.out.println(date);
  }

  public static void m1(Date date) {
    date = new Date();
  }
}

(b)
import java.util.Date;

public class Test {
  public static void main(String[] args) {
    Date date = new Date(1234567);
    m1(date);
    System.out.println(date.getTime());
  }

  public static void m1(Date date) {
    date = new Date(7654321);
  }
}

(c)
import java.util.Date;

public class Test {
  public static void main(String[] args) {
    Date date = new Date(1234567);
    m1(date);
    System.out.println(date.getTime());
  }

  public static void m1(Date date) {
    date.setTime(7654321);
  }
}

(d)
import java.util.Date;
```

```
public class Test {
  public static void main(String[] args) {
    Date date = new Date(1234567);
    m1(date);
    System.out.println(date.getTime());
  }

  public static void m1(Date date) {
    date = null;
  }
}
```

(a) null
(b) 1234567
(c) 7654321
(d) 1234567

Note that Java passes the value to an argument when invoking a method. In (a) and (b), the reference of a Date object is passed to a Date parameter in method m1. The method assigns a Date object to variable date. The variable date is changed in method m1, but it does not change date in the main method. Same is as in (d).
In (c), the main method invokes m1 by passing the reference of the Date object. Method m1 sets a new time in the Date object. Since the reference variable date in both the main method and m1 point to the same Date object, a new time is set in method m1 using setTime(7654321), the getTime() is used to retrieve this time from the same object in the main method.

Hide Answer

Section 9.11

▼9.11.1

What is wrong in the following code?

```
1  public class Test {
2    public static void main(String[] args) {
3      java.util.Date[] dates = new java.util.Date[10];
4      System.out.println(dates[0]);
5      System.out.println(dates[0].toString());
6    }
7  }
```

(Line 4 prints null since dates[0] is null. Line 5 causes a NullPointerException since it invokes toString() method from the null reference.)

Hide Answer

Section 9.12

▼9.12.1

If a class contains only private data fields and no setter methods, is the class immutable?

Not necessarily. To be immutable, the class must also contain no getter methods that would return a reference to a mutable data field object.

Hide Answer

▼9.12.2

If all the data fields in a class are private and of primitive types, and the class doesn't contain any setter methods, is the class immutable?

Yes

Hide Answer

▼9.12.3

Is the following class immutable?

```
public class A {
  private int[] values;

  public int[] getValues() {
    return values;
  }
}
```

No, because values is a reference type.

Hide Answer

Section 9.13

▼9.13.1

What is the output of the following program?

```
public class Test {
  private static int i = 0;
  private static int j = 0;

  public static void main(String[] args) {
    int i = 2;
    int k = 3;

    {
      int j = 3;
      System.out.println("i + j is " + i + j);
    }

    k = i + j;
    System.out.println("k is " + k);
    System.out.println("j is " + j);
  }
}
```

i + j is 23 (because i (value of 2) is concatenated with string i + j is, then j (value of 3) is concatenated with string i + j is 2.)

```
k is 2
j is 0
```

Hide Answer

Section 9.14

▼9.14.1

Describe the role of the this keyword.

this refers to the object itself

Hide Answer

▼9.14.2

What is wrong in the following code?

```
 1  public class C {
 2    private int p;
 3
 4    public C() {
 5      System.out.println("C's no-arg constructor invoked");
 6      this(0);
 7    }
 8
 9    public C(int p) {
10      p = p;
11    }
12
13    public void setP(int p) {
14      p = p;
15    }
16  }
```

Swap line 5 with line 6
Lines 10, 14, should be this.p = p;

Hide Answer

▼9.14.3

What is wrong in the following code?

```
public class Test {
  private int id;

  public void m1() {
    this.id = 45;
  }

  public void m2() {
    Test.id = 45;
  }
}
```

Test.id = 45; This is wrong, since id is an instance member and cannot be accessed from a class.

Hide Answer