Chapter 8 - Review Questions

Sections 8.2–8.5

8.1 Describe the relationship between an object and its defining class. How do you define a class? How do you declare an object reference variable? How do you create an object? How do you declare and create an object in one statement?

Class is the blueprint of an object. What we want, how we want something in a object, we have to write in class file.

```
// Class definition Example
class ExampleClass
{
// Class data fields and methods
}

// Class reference variable declaration
ExampleClass exampleObject;

// Creating an object
exampleObject = new ExampleClass();

// Declare and create an object in one statement.
ExampleClass exampleObject = new ExampleClass();
```

8.2 What are the differences between constructors and methods?

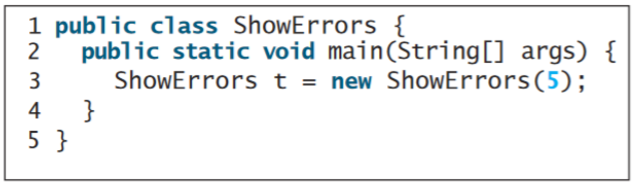
Constructors are methods too. But they are specific type of methods. They are only use for creating new objects.

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

Array is an object type value in java. It stores reference of variables therefore, it can keep containing both object type and primitive type values.
Default value of elements of an array like this;

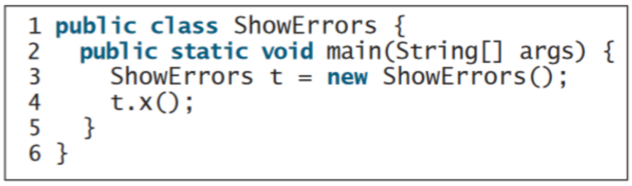
- boolean -> false
- int -> 0
- double -> 0.0
- String -> null
- Objects -> null

8.4 What is wrong with the following program?



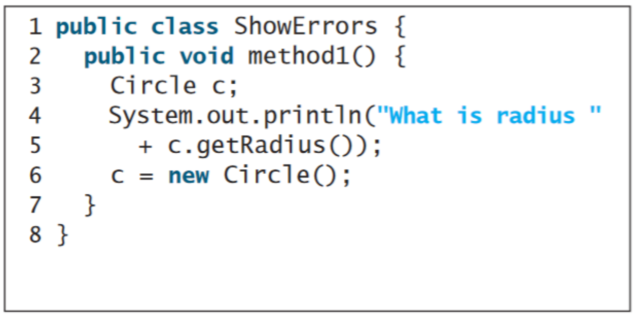
```java
public class ShowErrors {
  public static void main(String[] args) {
     ShowErrors t = new ShowErrors(5);
  }
}
```

(a)



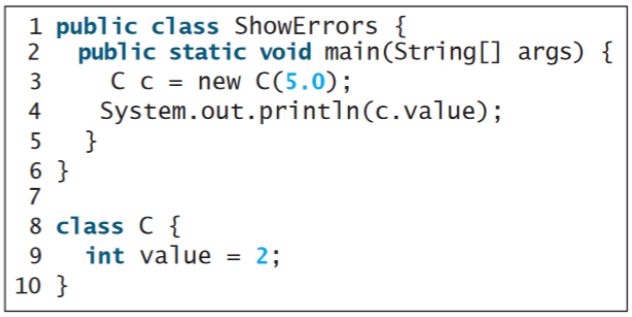
```java
public class ShowErrors {
  public static void main(String[] args) {
     ShowErrors t = new ShowErrors();
     t.x();
  }
}
```

(b)



```java
public class ShowErrors {
  public void method1() {
    Circle c;
    System.out.println("What is radius "
      + c.getRadius());
    c = new Circle();
  }
}
```

(c)



```java
public class ShowErrors {
  public static void main(String[] args) {
     C c = new C(5.0);
    System.out.println(c.value);
  }
}

class C {
  int value = 2;
}
```

(d)

a) There isn't any constructor that can take argument in class definition
b) There isn't any method called *x* in class definition.
c) There isn't any class definition called *Circle* in program.
d) There isn't any constructor that can take argument in class definition

---

8.5 What is wrong in the following code?

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

class A {
  String s;

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

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

The program has an constructor that can take argument but hasn't any default (*no argument taken*) constructor. In java if you have constructor that can take argument you have to create default constructor too.

8.6 What is the printout of the following code?

```
public class Foo {
  private boolean x;

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

Printout will be *false*. Because default value of boolean in java is false.

Section 8.6

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

Using *Date* class in java we can easily find current time. Method in below gives to us current time.
`date.toString()`

8.8 How do you create a JFrame, set a title in a frame, and display a frame?

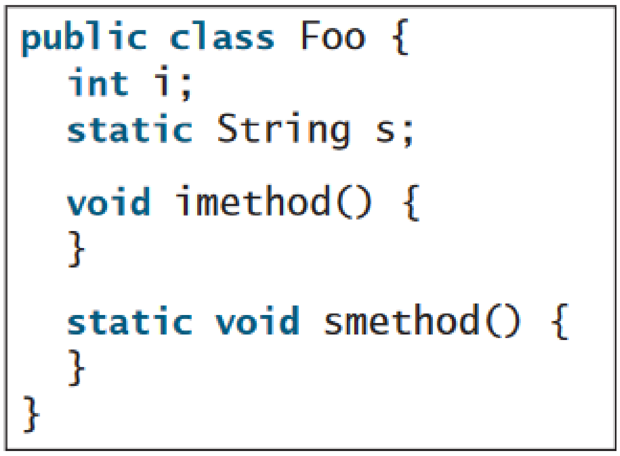
```
// First we have to JFrane library
import javax.swing.JFrame;

public class Test
{
        public static void main( String[] args )
        {
                // We can create a JFrame calling it's constructor
                JFrame exampleFrame = new JFrame();

                // We can set a title in JFrame with *setTitle* method
                exampleFrame.setTitle("Window");

                // Set proper size for frame
                exampleFrame.setSize( 1000,1000);

                // Make frame visible
                exampleFrame.setVisible(true);
        }
}
```

8.9 Which packages contain the classes Date, JFrame, JOptionPane, System, and

Math?
Date -> java.util
JFrame, JOptionPane -> javax.swing
System, Math -> java.lang

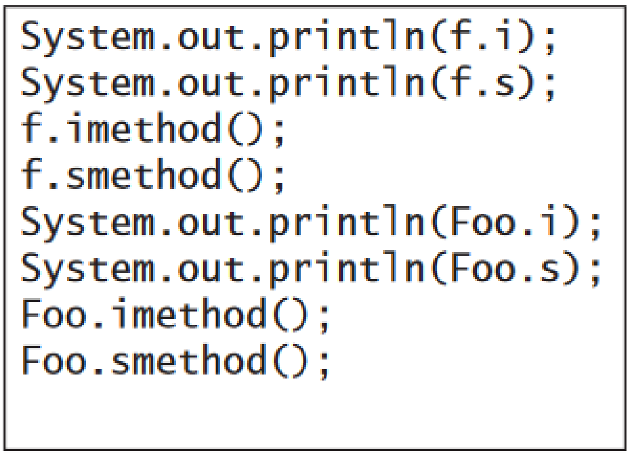
8.10 Suppose that the class Foo is defined in (a). Let f be an instance of Foo. Which of

the statements in (b) are correct?



```
public class Foo {
  int i;
  static String s;

  void imethod() {
  }

  static void smethod() {
  }
}
```

(a)

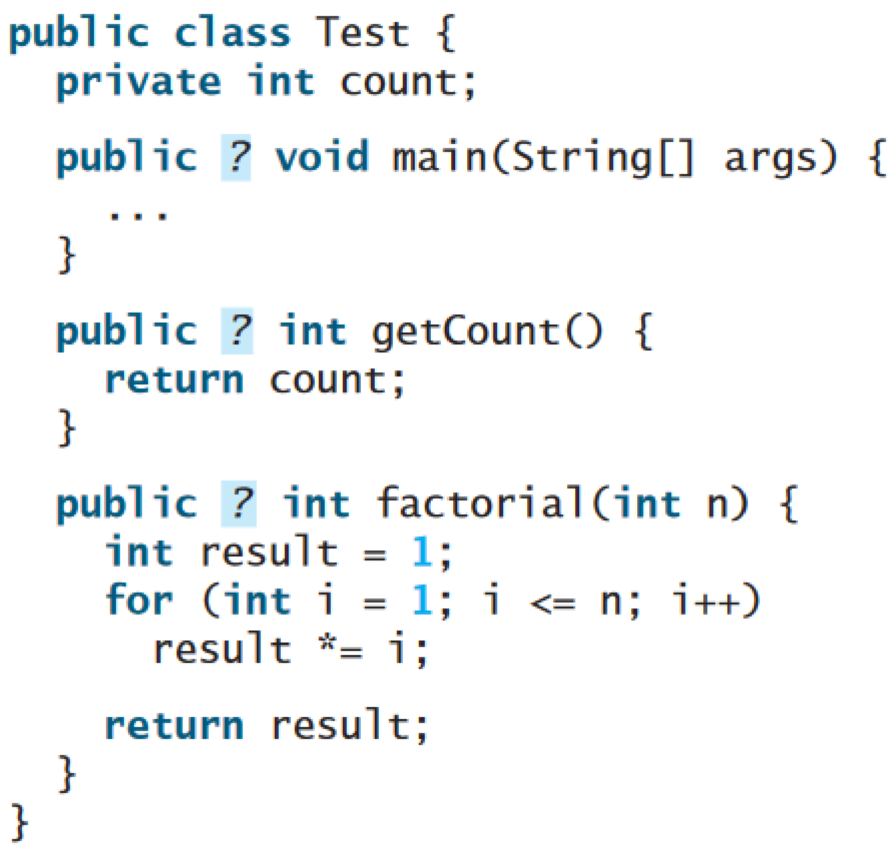


```
System.out.println(f.i);
System.out.println(f.s);
f.imethod();
f.smethod();
System.out.println(Foo.i);
System.out.println(Foo.s);
Foo.imethod();
Foo.smethod();
```

(b)

`System.out.println(Foo.i);` is false because non-static variable cannot be referenced from a static context *(main method)* and `Foo.imethod();` is also false because non-static method cannot be referenced from a static context *(main method)*.
Others are true.

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



```
public class Test {
  private 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;
  }
}
```

Just `public static void main( String[] args )`

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

```java
public class Foo {
  public static void main(String[] args) {
    method1();
  }

  public void method1() {
    method2();
  }

  public static void method2() {
    System.out.println("What is radius " + c.getRadius());
  }

  Circle c = new Circle();
}
```

We can't reference an instance method or an instance variable from a static method. But we can reference a static method or a static variable from an instance method.
Calling method1 from *static* main method is cause an error. On the other hand, creating a Circle object after calling circle object related method cause an error too.

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

Accessor method useful for when we need access a private value; mutator method useful for when we set a new value for a private value.
The naming convention for accessor is get*Value*; for mutator is set*Value*.

8.14 What are the benefits of data-field encapsulation?

1. Protecting data from unrelated people.
2. Making easy to develop and debug the code.

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

```java
public class Circle {
  private double radius = 1.0;

  /** 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);
  }
}
```

It won't cause any problem. When we create an object, we create an instance of its class. Therefore, instance object contain all variables and methods in its class

---

8.16 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 program:

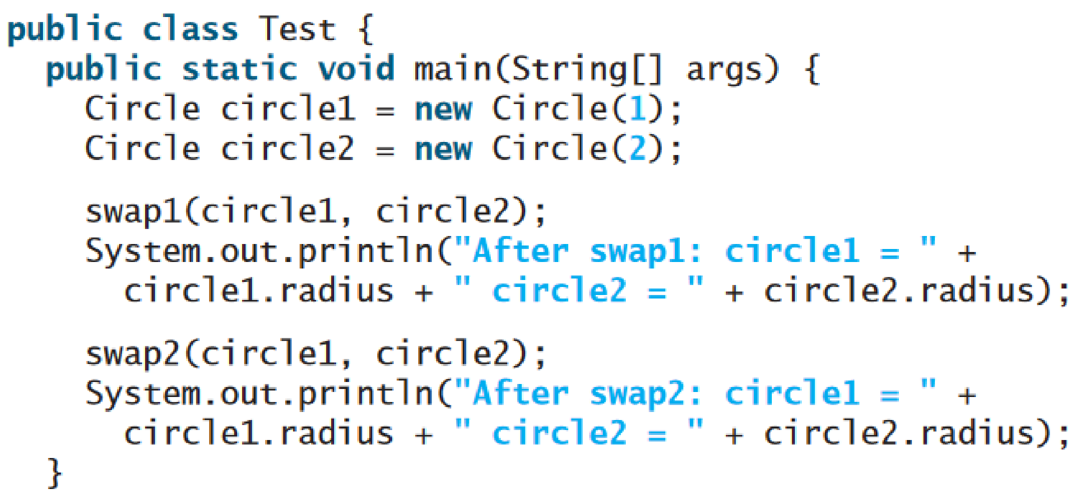
```java
public class Test {
  public static void main(String[] args) {
    Count myCount = new Count();
    int times = 0;

    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++;
  }
}
```

```java
public class Count {
  public int count;

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

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

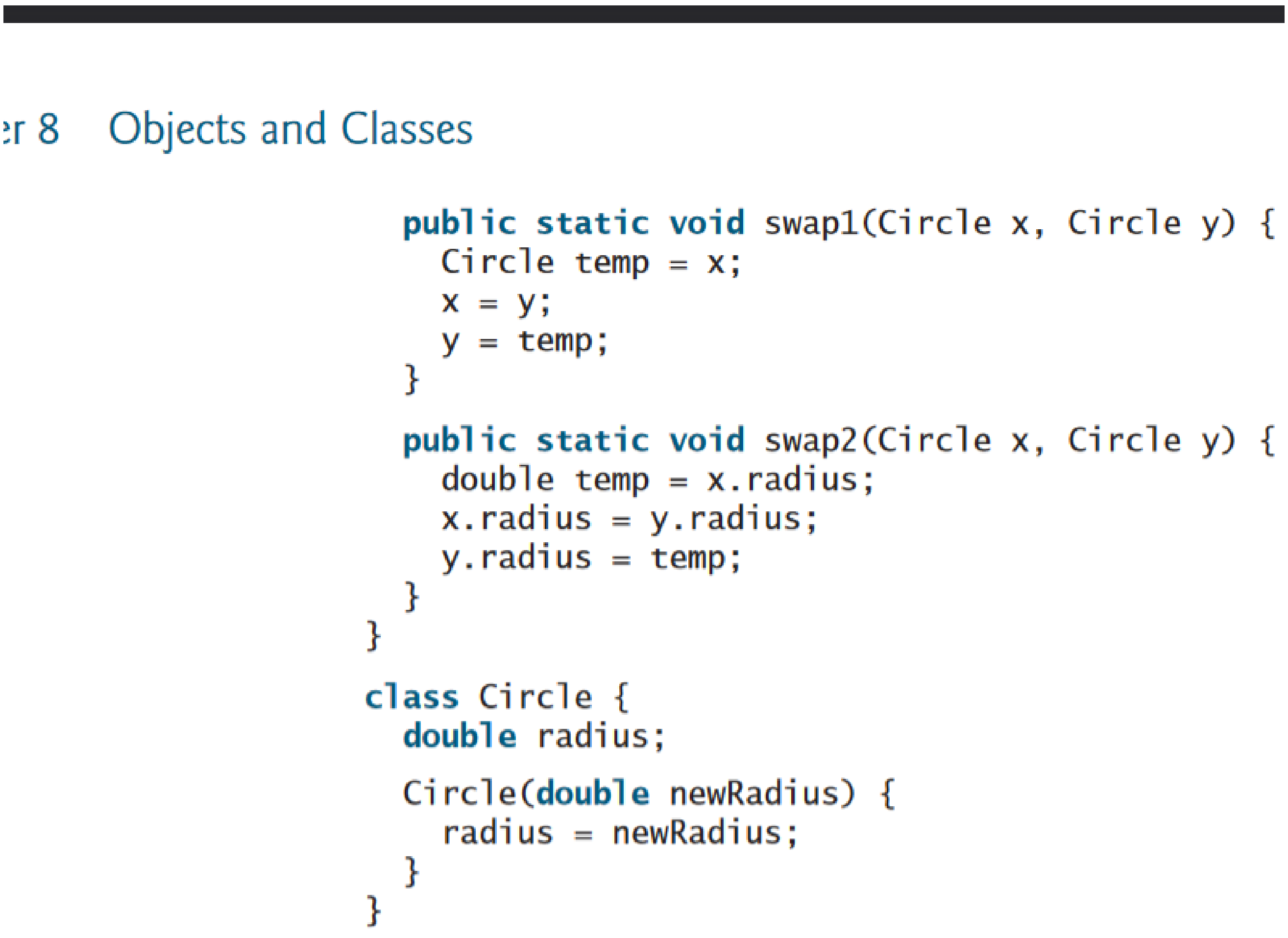
When we pass reference type an argument to a method, we actually pass the address of the variable. In this way, modification at the variable in the method also exist when the method ends. For primitive type of argument, this is not going to to happen. All the modification in the method will be gone when method ends.

```
//Output:

count is 101
times is 0
```

8.17 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);
  }
```

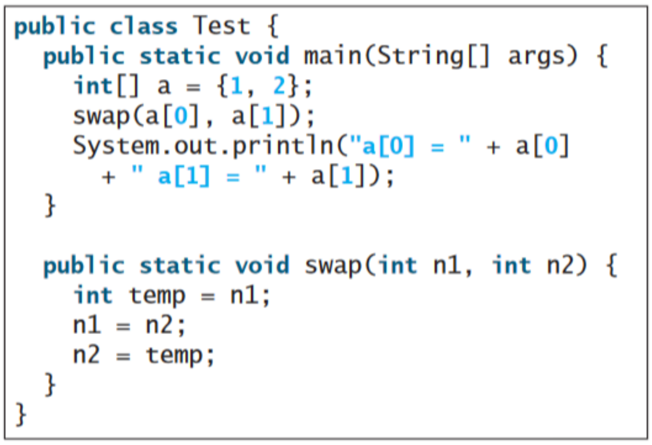


er 8 Objects and Classes

```
  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;
  }
}
```

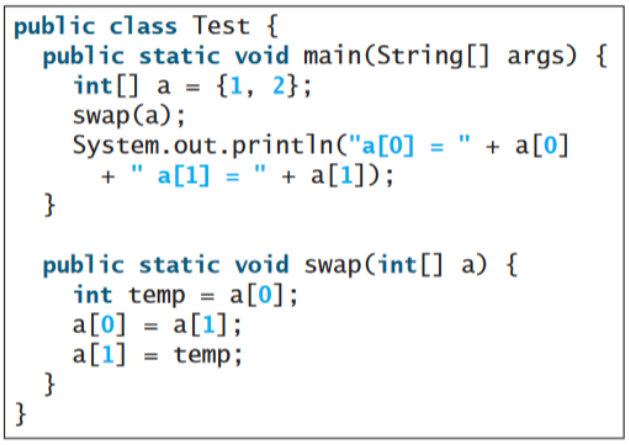
```
// Output:
After swap1: circle1 = 1.0 circle2 = 2.0
After swap2: circle1 = 2.0 circle2 = 1.0
```

8.18 Show the printout of the following code:



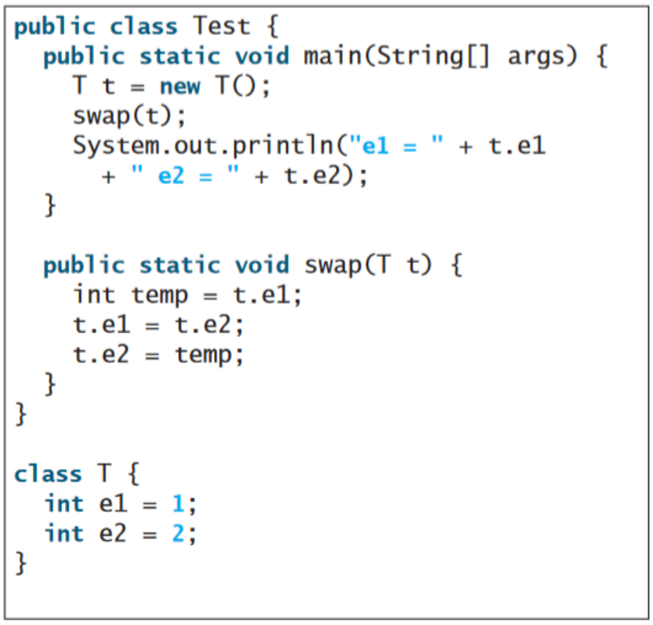
```java
public class Test {
  public static void main(String[] args) {
    int[] a = {1, 2};
    swap(a[0], a[1]);
    System.out.println("a[0] = " + a[0]
      + " a[1] = " + a[1]);
  }

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

(a)



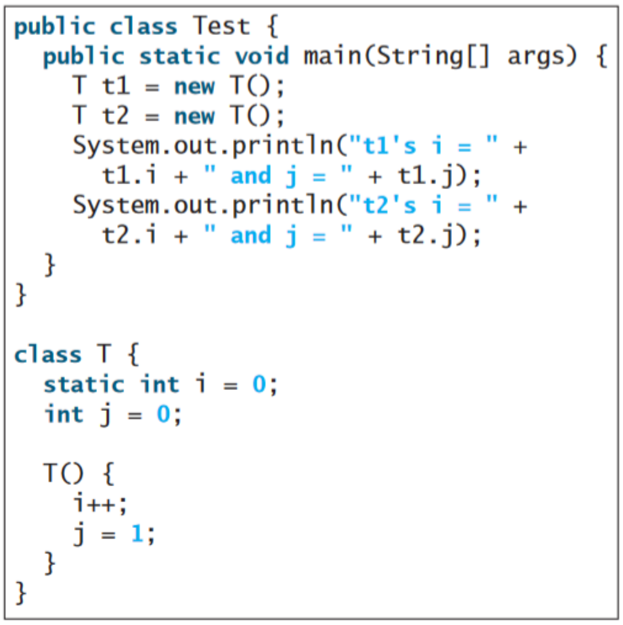
```java
public class Test {
  public static void main(String[] args) {
    int[] a = {1, 2};
    swap(a);
    System.out.println("a[0] = " + a[0]
      + " a[1] = " + a[1]);
  }

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

(b)



```java
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;
}
```

(c)



```java
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;
  }
}
```

(d)

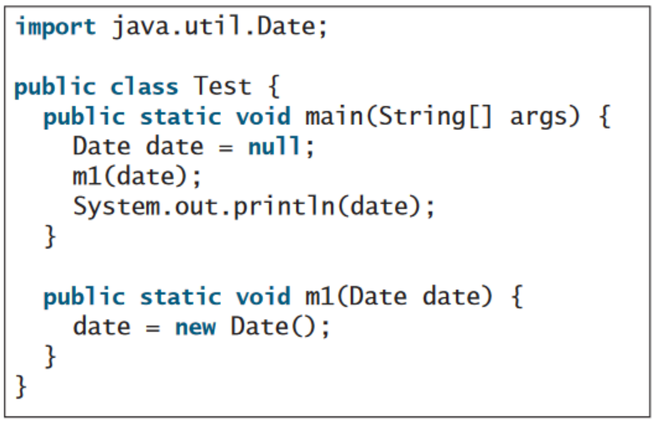
```
// (a) Output
a[0] = 1 a[1] = 2
```

```
// (b) Output
a[0] = 2 a[1] = 1
```

```
// (c) Output
e1 = 2 e2 = 1
```

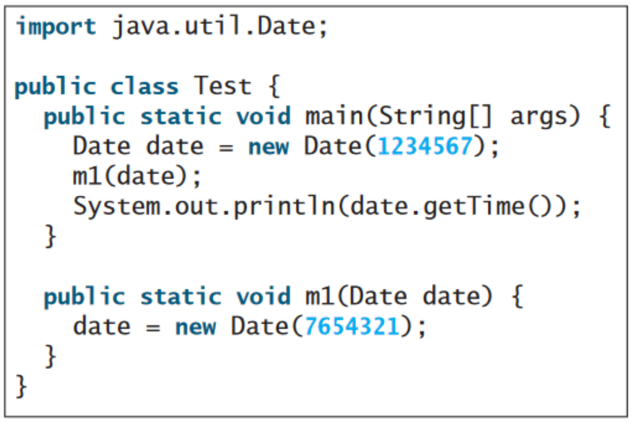
```
// (d) Output
t1's i = 2 and j = 1
t2's i = 2 and j = 1
```

8.19 What is the output of the following program?



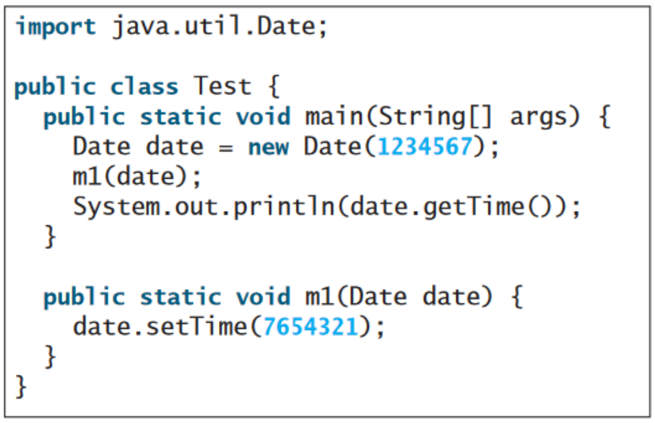
```java
import java.util.Date;

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();
  }
}
```

(a)



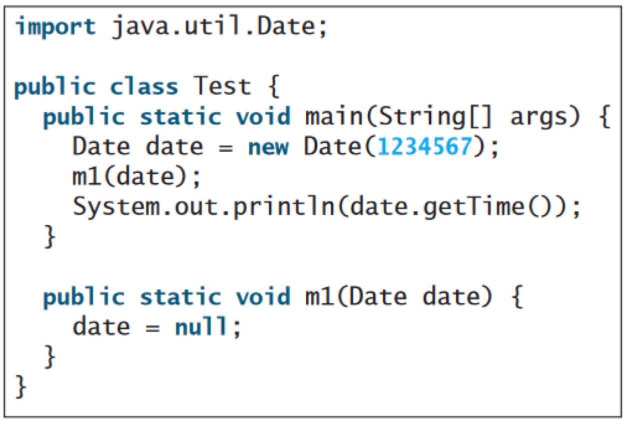
```java
import java.util.Date;

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);
  }
}
```

(b)



```java
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);
  }
}
```

(c)



```java
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;
  }
}
```

(d)

```
// (a) Output
null
```

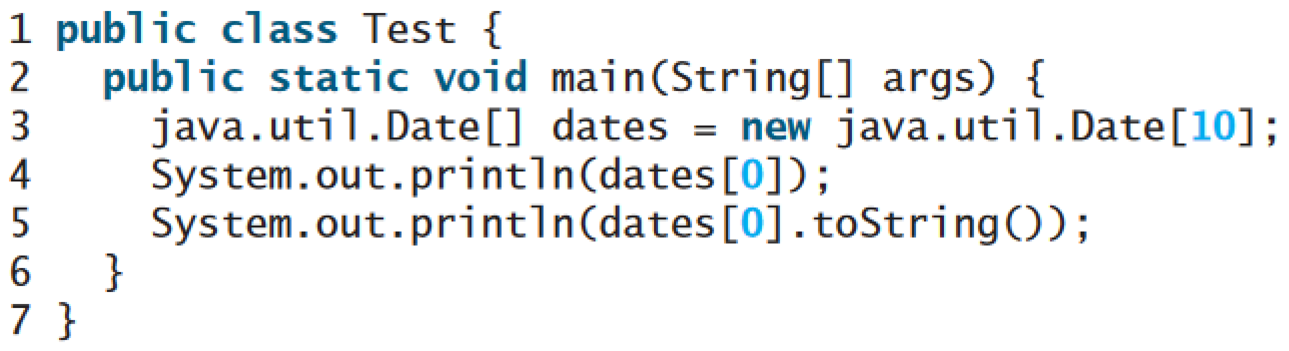
```
// (b) Output
1234567
```

```
// (c) Output
7654321
```

```
// (d) Output
1234567
```

Section 8.11

8.20 What is wrong in the following code?



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

dates arrays' elements are not initialized. So, there is nothing where pointer point. Therefore, compiler gives NullPointerException.