

# Multiple Choice

## Object Methods

01. Consider the following statement:

*Blank is a computer science tool that involves using program features without knowledge of how the program features are implemented.*

This statement explains

- (A) inheritance.
- (B) encapsulation.
- (C) composition.
- (D) information hiding.
- (E) polymorphism.

02. An object is a

- (A) single instance of a given data structure template.
- (B) collection of primitive data types.
- (C) user-defined data type.
- (D) data structure template or blue print.

03. Examples of class methods are

- (A) **Math** methods.
- (B) **Bank** methods.
- (C) **Random** and **DecimalFormat** methods.
- (D) both B and C.

04. Examples of object methods are

- (A) **Math** methods.
- (B) **Bank** methods.
- (C) **Random** and **DecimalFormat** methods
- (D) both B and C.

05. A class method is called by using

- (A) the method identifier only.
- (B) the class identifier only.
- (C) the class identifier, followed by a period and the method identifier.
- (D) an object identifier, followed by a period and the method identifier.

06. An object method (or instance method) is called by using

- (A) the method identifier only.
- (B) the object identifier only.
- (C) the class identifier, followed by a period and the method identifier.
- (D) an object identifier, followed by a period and the method identifier.

07. The Java keyword **new** is used to construct

- (A) objects only.
- (B) classes only.
- (C) objects and classes.
- (D) neither objects nor classes.

08. By Java program writing convention, which of the following is true?

- (A) Object identifiers start with a capital letter and class identifiers start with a lower-case letter.
- (B) Object identifiers start with a lower-case letter and class identifiers start with a capital letter.
- (C) Object identifiers start with a capital letter and class identifiers start with a capital letter.
- (D) Object identifiers start with a lower-case letter and class identifiers start with a lower-case letter.

09. The methods in the **Math** class are

- (A) class methods.
- (B) object methods.
- (C) expression methods.
- (D) variable methods.

10. The methods in the **Bank** class are

- (A) class methods.
- (B) object methods.
- (C) expression methods.
- (D) variable methods.

11. The methods in the **Random** class are

- (A) class methods.
- (B) object methods.
- (C) expression methods.
- (D) variable methods.

12. The methods in the **DecimalFormat** class are

- (A) class methods.
- (B) object methods.
- (C) expression methods.
- (D) variable methods.

For questions 13-20 use the following **Bank** class information. It contains the headings of the methods in the **Bank** class, along with a description.

**public Bank()**

// default constructor starts checking and savings account with zero dollars.

**public Bank(double c, double s)**

// constructor creates an object with **c** dollars in checking and **s** dollars in savings.

**public double getChecking()**

// returns the checking account balance

**public double getSavings()**

// returns the savings account balance

**public double getCombined()**

// returns the combined balance of the checking and savings account

**public void changeChecking(double amount)**

// alters the balance of the checking account by the **amount** parameter

**public void changeSavings(double amount)**

// alters the balance of the savings account by the **amount** parameter

**public void closeChecking()**

// alters the checking account balance to zero

**public void closeSavings()**

// alters the savings account balance to zero

13. Access to methods of the **Bank** class requires

- (A) using a statement, like **Bank.getSavings();**
- (B) using a statement, like **Bank.getSavings;**
- (C) the creation of one or more **Bank** objects.
- (D) using the **get** method.

14. A class is \_\_\_\_\_ .

- (A) a data type
- (B) a variable
- (C) a constant
- (D) another name for an object

15. An object is \_\_\_\_\_ .

- (A) a data type
- (B) a variable
- (C) a constant
- (D) another name for a class

16. The **new** keyword must be used with

- (A) every class method call.
- (B) every object method call.
- (C) the construction of each object.
- (D) the construction of each class.

17. What is the output of the following program segment?

```
Bank tom;  
tom = new Bank();  
Bank sue;  
sue = new Bank();  
tom.changeChecking(1000);  
sue.changeChecking(1500);  
System.out.println("sue: " + sue.getSavings());  
System.out.println("tom: " + tom.getSavings());
```

(A) tom: 1000.0  
sue: 1500.0

(B) sue: 1500.0  
tom: 1000.0

(C) sue: 0.0  
tom: 0.0

(D) Error message

18. What is the output of the following program segment?

```
Bank tom;  
tom = new Bank(7500.0, 5000.0);  
Bank sue;  
sue = new Bank(4000.0, 3500.0);  
System.out.println("tom: " + tom.getChecking() + " " + tom.getSavings());  
System.out.println("sue: " + sue.getChecking() + " " + sue.getSavings());  
tom.closeChecking(); tom.closeSavings();  
sue.closeChecking(); sue.closeSavings();
```

(A) sue: 7500.0 5000.0  
tom: 4000.0 3500.0

(B) tom: 7500.0 5000.0  
sue: 4000.0 3500.0

(C) sue: 0.0  
tom: 0.0

(D) Error message

19. Consider the two segments below.  
Do both segments properly construct a **tom** object?

**// segment 1**

**Bank tom;**

**tom = new Bank(7500, 5000);**

**// segment 2**

**Bank tom = new Bank(7500, 5000);**

- (A) Segment 1 is correct and segment 2 is not correct.  
(B) Segment 1 is incorrect and segment 2 is correct.  
(C) Both segments are incorrect.  
(D) Both segments are correct.

20. Consider the two segments below.  
Do both segments properly construct a **tom** object?

**// segment 1**

**Bank tom;**

**tom = new Bank(7,500.0, 5,000.0);**

**// segment 2**

**Bank tom = new Bank(7,500.0, 5,000.0);**

- (A) Segment 1 is correct and segment 2 is not correct.  
(B) Segment 1 is incorrect and segment 2 is correct.  
(C) Both segments are incorrect.  
(D) Both segments are correct.

21. The **Random** class is found inside the \_\_\_\_\_ package.

- (A) **util.java**
- (B) **java.util**
- (C) **java.util.Random**
- (D) None of the above

22. Assume that **rand** is an object of the **Random** class. Which of the following statements controls the generation of the random numbers, such that each execution generates the same sequence of numbers?

- (A) **rand.setSeed(3333);**
- (B) **rand.setSequence(3333);**
- (C) **Random rand = new Random(3333);**
- (D) Both A and C

23. Assume that **rand** is an object of the **Random** class.  
Which of the following statements generates a random number in the [0..100] range?

- (A) **int number = rand.nextInt(100) + 0;**
- (B) **int number = rand.nextInt(101);**
- (C) **int number = rand.nextInt(0) + 100;**
- (D) **int number = rand.nextInt(0) + 101;**

24. Assume that **rand** is an object of the **Random** class.  
Which of the following statements generates a random number in the [1..1000] range?

- (A) **int number = rand.nextInt(1000) + 1;**
- (B) **int number = rand.nextInt(1000);**
- (C) **int number = rand.nextInt(1001);**
- (D) **int number = rand.nextInt(1) + 1000;**



25. Assume that **rand** is an object of the **Random** class.  
Which of the following statements generates a random number in the [200..600] range?

- (A) `int number = rand.nextInt(200) + 600;`
- (B) `int number = rand.nextInt(600) + 200;`
- (C) `int number = rand.nextInt(400) + 200;`
- (D) `int number = rand.nextInt(401) + 200;`

26. Assume that **rand** is an object of the **Random** class.  
Which of the following statements generates a random number in the [41..101] range?

- (A) `int number = rand.nextInt(41) + 101;`
- (B) `int number = rand.nextInt(101) + 41;`
- (C) `int number = rand.nextInt(61) + 41;`
- (D) `int number = rand.nextInt(60) + 41;`

27. Assume that **rand** is an object of the **Random** class.  
Which of the following statements generates a random number in the [-41..101] range?

- (A) `int number = rand.nextInt(143) + 41;`
- (B) `int number = rand.nextInt(143) - 41;`
- (C) `int number = rand.nextInt(101) - 41;`
- (D) `int number = rand.nextInt(41) + 101;`

28. Assume that **rand** is an object of the **Random** class.  
Which of the following statements generates a random number in the [-101..-41] range?

- (A) `int number = rand.nextInt(61) - 101;`
- (B) `int number = rand.nextInt(61) - 41;`
- (C) `int number = rand.nextInt(-41) - 101;`
- (D) `int number = rand.nextInt(-101) - 41;`

29. Assume that `rand` is an object of the **Random** class.  
Which of the following ranges is generated by this statement:  
**`int number = rand.nextInt(1201) + 400; ?`**

- (A) [400..1200]
- (B) [400..1201]
- (C) [400..1600]
- (D) [400..1601]

30. Assume that `rand` is an object of the **Random** class.  
Which of the following ranges is generated by this statement:  
**`int number = rand.nextInt(72) + 57;`**

- (A) [57..72]
- (B) [57..127]
- (C) [57..128]
- (D) [57..129]

31. Assume that `rand` is an object of the **Random** class.  
Which of the following ranges is generated by this statement:  
**`int number = rand.nextInt(250) - 125;`**

- (A) [-125..125]
- (B) [-124..124]
- (C) [-125..124]
- (D) [-125..126]
- (E) [-125..250]

32. Assume that `rand` is an object of the **Random** class.  
Which of the following statements generates a random character in the ['A'..'Z'] range?

- (A) **`int letter = rand.nextInt(65) + 26;`**
- (B) **`int letter = rand.nextInt(26) + 65;`**
- (C) **`char letter = (char) rand.nextInt(65) + 26;`**
- (D) **`char letter = (char) rand.nextInt(26) + 65;`**

33. What is the output of the following program?

```
import java.util.Random;

public class Question33
{
    public static void main(String args[ ])
    {
        Random rand = new Random();
        rand.setSeed(100);
        System.out.println(rand.nextInt(900) + 100);
        System.out.println(rand.nextInt(900) + 100);
        System.out.println(rand.nextInt(900) + 100);
        System.out.println(rand.nextInt(900) + 100);
        System.out.println(rand.nextInt(900) + 100);
        System.out.println(rand.nextInt(900) + 100);
        System.out.println(rand.nextInt(900) + 100);
        System.out.println(rand.nextInt(900) + 100);
        System.out.println(rand.nextInt(900) + 100);
        System.out.println(rand.nextInt(900) + 100);
    }
}
```

- (A) 10 different integers in the [100..999] range
- (B) 10 identical integers in the [100..900] range
- (C) 10 different integers in the [100..900] range
- (D) 10 identical integers in the [100..1000] range

34. What is the output of the following program?

```
import java.util.Random;

public class Question34
{
    public static void main(String args[ ])
    {
        Random rand = new Random();
        rand.setSeed(100);
        System.out.println(rand.nextDouble());
        System.out.println(rand.nextDouble());
        System.out.println(rand.nextDouble());
        System.out.println(rand.nextDouble());
        System.out.println(rand.nextDouble());
    }
}
```

- (A) 5 double numbers in an unknown range.
- (B) 5 double numbers *greater-or-equal to zero and less-than 100*.
- (C) 5 double numbers *greater-or-equal to zero and less-than 1*.
- (D) 5 double numbers *greater-or-equal to 100*.

35. The **Math.random** method generates a random number **x**, such that

- (A)  $0 \leq x < 1$
- (B)  $0 < x < 1$
- (C)  $0 < x \leq 1$
- (D)  $0 < x$

36. Assume that **rand** is an object of the **Random** class.  
Numbers generated by a call to **Math.random()** are in the same range as numbers called by

- (A) **rand.nextDouble();**
- (B) **rand.nextInt();**
- (C) **rand.nextDouble(0,1);**
- (D) **rand.nextSeed();**

37. Consider the following program segment.

```
for (int k = 1; k <= 10; k++)  
{  
    double var1 = Math.random() * 100;  
    int var2 = (int) var1 + 10;  
    System.out.print(var2 + " ");  
}
```

Which of the following are possible outputs of executing the program segment?

- (A) 46 78 11 18 99 87 42 21 43 78
- (B) 91 100 88 60 85 71 22 37 75 49
- (C) 10 17 46 56 18 99 21 44 89 75
- (D) All of the above.

38. What is the range of possible random integers generated by the program segment in question 37?

- (A) [10 . . . 100]
- (B) [10 . . . 109]
- (C) [11 . . . 109]
- (D) [10 . . . 110]

39. Which of the following program segments generates a random integer in the range [1000 . . . 9999] ?

(A)  
`int range = 9999 - 1000 + 1;  
int randInt = (int) Math.random() * range + 1000;`

(B)  
`int range = 9999 - 1000 + 1;  
int randInt = (int) (Math.random()) * range + 1000;`

(C)  
`int randInt = (int) (Math.random() * 9999) + 1000;`

(D)  
`int range = 9999 - 1000 + 1;  
int randInt = (int) (Math.random() * range) + 1000;`

40. The **DecimalFormat** class is found inside the \_\_\_\_\_ package.

- (A) **text.java**
- (B) **java.text**
- (C) **java.util**
- (D) None of the above

41. The **DecimalFormat** class has the ability to

- (A) round off decimal numbers to a specified number of digits.
- (B) format numbers with leading zeroes.
- (C) format numbers with trailing zeroes.
- (D) do all of the above.

42. The kind of output created by an object of the **DecimalFormat** class is determined by

- (A) the type of parameter used with the construction of a new **DecimalFormat** object.
- (B) using the **format** method.
- (C) using the **output** method.
- (D) all of the above.

43. What is the output of this program?

```
import java.text.DecimalFormat;

public class Question43
{
    public static void main (String args[ ])
    {
        DecimalFormat output = new DecimalFormat("000,000,000");
        System.out.println(output.format(1));
        System.out.println(output.format(12));
        System.out.println(output.format(123));
        System.out.println(output.format(1234));
        System.out.println(output.format(12345));
        System.out.println(output.format(123456));
        System.out.println(output.format(1234567));
        System.out.println(output.format(12345678));
        System.out.println(output.format(123456789));
    }
}
```

(A) 1  
12  
123  
1,234  
1,234,5  
1,234,56  
1,234,567  
1,234,567,8  
1,234,567,89

(B) 1  
12  
123  
1,234  
12,345  
123,456  
1,234,567  
12,345,678  
123,456,789

(C) 000,000,001  
000,000,012  
000,000,123  
000,001,234  
000,012,345  
000,123,456  
001,234,567  
012,345,678  
123,456,789

(D) 000000001  
000000012  
000000123  
000001234  
000012345  
000123456  
001234567  
012345678  
123456789

(E) Error Message

44. What is the output of this program?

```
import java.text.DecimalFormat;

public class Question44
{
    public static void main (String args[ ])
    {
        DecimalFormat output = new DecimalFormat("0000000");
        System.out.println(output.format(1));
        System.out.println(output.format(12));
        System.out.println(output.format(123));
        System.out.println(output.format(1234));
        System.out.println(output.format(12345));
        System.out.println(output.format(123456));
        System.out.println(output.format(1234567));
        System.out.println(output.format(12345678));
        System.out.println(output.format(123456789));
    }
}
```

(A) 1  
12  
123  
1234  
12345  
123456  
1234567  
12345678  
123456789

(B) 1  
12  
123  
1234  
12345  
123456  
1234567  
12345678  
123456789

(C) 000000001  
000000012  
000000123  
000001234  
000012345  
000123456  
001234567  
012345678  
123456789

(D) 0000001  
0000012  
0000123  
0001234  
0012345  
0123456  
1234567  
12345678  
123456789

(E) Error Message



45. What is the output of this program?

```
import java.text.DecimalFormat;

// Math.PI will return a value of 3.14159265358979

public class Question45
{
    public static void main (String args[ ])
    {
        DecimalFormat output1 = new DecimalFormat("$0.00");
        DecimalFormat output2 = new DecimalFormat("00.0000");
        DecimalFormat output3 = new DecimalFormat("0,000,000");
        double millionPI = 1000000.0 * Math.PI;
        System.out.println(output1.format(Math.PI));
        System.out.println(output2.format(Math.PI));
        System.out.println(output3.format(millionPI));
    }
}
```

- |                                    |                                    |                                  |                                    |
|------------------------------------|------------------------------------|----------------------------------|------------------------------------|
| (A) \$3.14<br>03.1416<br>3,141,593 | (B) \$3.14<br>03.1416<br>3,141,592 | (C) 3.14<br>03.1416<br>3,141,593 | (D) 3,141,593<br>03.1416<br>\$3.14 |
| (E) Error Message                  |                                    |                                  |                                    |

46. A *wrapper* class creates objects capable of
- (A) wrapping one class inside another class.
  - (B) wrapping multiple classes inside one object.
  - (C) storing primitive data values like **int** and **double**.
  - (D) combining multiple objects inside a single class.
47. Which of the following statements constructs an **Integer** object correctly?
- (A) **Integer obj = new Integer(100);**
  - (B) **Integer obj = new Integer();**
  - (C) **Integer obj = new Integer(int 100);**
  - (D) All of the above.
48. A feature in Java version 5.0 and later, which automatically handles wrapping, but is not used on the AP Computer Science Examination is called
- (A) auto-wrapping.
  - (B) boxing.
  - (C) box-wrapping.
  - (D) auto-boxing.
49. `Integer.MAX_VALUE` and `Integer.MIN_VALUE` are examples of
- (A) object methods.
  - (B) class methods.
  - (C) constant fields or attributes.
  - (D) variable fields or attributes.
50. In the statement **int temp = Integer.MAX\_VALUE + 1;** variable **temp** stores
- (A) a negative integer value.
  - (B) an integer value larger than the maximum Integer value.
  - (C) the maximum Integer value, since the larger value is not accepted.
  - (D) a real number value, which can handle larger values than integers.
-

51. **drawPolygon** & **fillPolygon** are both methods of the \_\_\_\_\_ class.

- (A) Math
  - (B) Random
  - (C) DecimalFormat
  - (D) Polygon
  - (E) Graphics
- 

52. **nextInt** & **nextDouble** are both methods of which classes?

- (A) Math and Random
  - (B) Random and Scanner
  - (C) DecimalFormat and Color
  - (D) Polygon and Graphics
  - (E) Graphics and Scanner
- 

53. Your monitor can display over 16 million different colors which are created by combining three primary colors. Which of the following is NOT one of those three primary colors?

- (A) red
  - (B) yellow
  - (C) green
  - (D) blue
- 

54. Assume **g** is an object of the **Graphics** class.  
Which of the following is using an *anonymous object*?

- (A) `g.setColor(green);`
  - (B) `g.setColor(Color.green);`
  - (C) `g.setColor(new Color(20,200,20));`
  - (D) `Color myGreen = new Color(20,200,20);`  
`g.setColor(myGreen);`
-

**In questions 55 through 58, refer to the following program segment which draws many lines all over an 800 x 600 Applet window.**

```
Random rndInt = new Random(12345);
for (int k = 1; k <= 1000; k++)
{
    int x1 = rndInt.nextInt(800);
    int y1 = rndInt.nextInt(600);
    int x2 = rndInt.nextInt(800);
    int y2 = rndInt.nextInt(600);
    g.drawLine(x1,y1,x2,y2);
}
```

---

55. How would you change this program to make the lines draw only in the TOP half of the screen?

- (A) Change the 1000 to 500
- (B) Change the 800's to 400's
- (C) Change the 600's to 300's
- (D) All of the above
- (E) Choices B and C only

---

56. How would you change this program to make the lines draw only in the LEFT half of the screen?

- (A) Change the 1000 to 500
- (B) Change the 800's to 400's
- (C) Change the 600's to 300's
- (D) All of the above
- (E) Choices B and C only

---

57. How would you change this program to make it display half as many LINES?

- (A) Change the 1000 to 500
  - (B) Change the 800's to 400's
  - (C) Change the 600's to 300's
  - (D) All of the above
  - (E) Choices B and C only
-

**In questions 55 through 58, refer to the following program segment which draws many lines all over an 800 x 600 Applet window.**

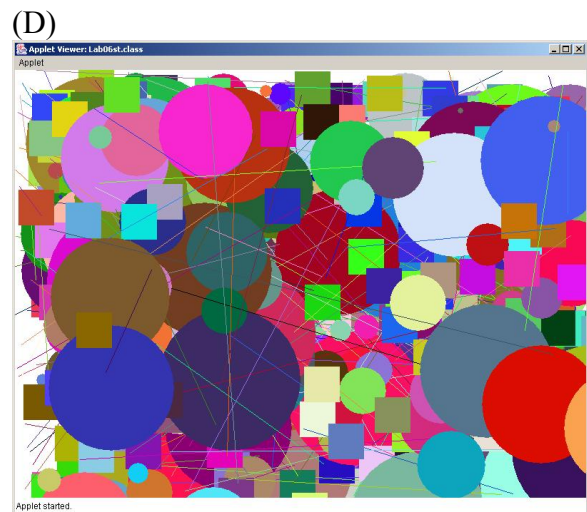
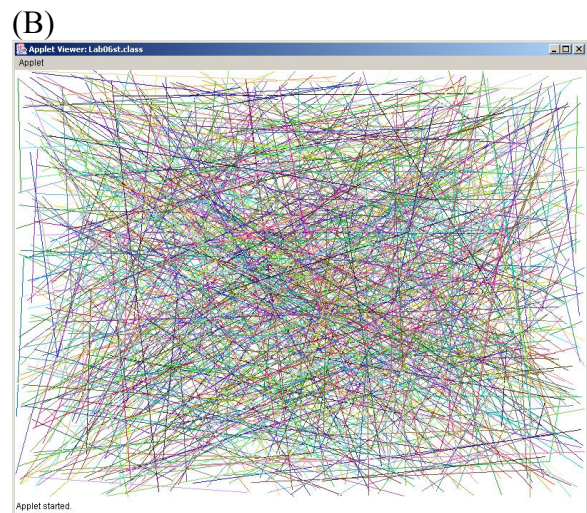
```
Random rndInt = new Random(12345);
for (int k = 1; k <= 1000; k++)
{
    int x1 = rndInt.nextInt(800);
    int y1 = rndInt.nextInt(600);
    int x2 = rndInt.nextInt(800);
    int y2 = rndInt.nextInt(600);
    g.drawLine(x1,y1,x2,y2);
}
```

- 
58. How would you change this program to make the lines draw only in the **TOP-LEFT QUARTER** of the screen?
- (A) Change the 1000 to 500
  - (B) Change the 800's to 400's
  - (C) Change the 600's to 300's
  - (D) All of the above
  - (E) Choices B and C only
- 
59. Which of the following program scenarios can cause a problem when using the *Scanner* class?
- (A) The program enters only numbers (ints or doubles)
  - (B) The program enters only Strings
  - (C) The program enters a number (int or double) BEFORE entering a string.
  - (D) The program enters a number (int or double) AFTER entering a string.
- 
60. Assume **g** is an object of the **Graphics** class.  
Which of the following will change the graphics color to a shade of **red**?
- (A) `g.setColor(new Color(200,0,0));`
  - (B) `g.setColor(new Color(200,200,200));`
  - (C) `g.setColor(new Color(0,0,0));`
  - (D) `g.setColor(new Color(255,255,255));`
-

61. What is the output of this program segment?

```
Random rnd = new Random(1234);
for (int count = 1; count <= 1000; count++)
{
    int red = rnd.nextInt(256);
    int green = rnd.nextInt(256);
    int blue = rnd.nextInt(256);
    g.setColor(new Color(red, green, blue));
    int x1 = rnd.nextInt(800);
    int y1 = rnd.nextInt(600);
    int x2 = rnd.nextInt(800);
    int y2 = rnd.nextInt(600);
    int diameter = rnd.nextInt(200);
    int shape = 0;

    switch (shape)
    {
        case 0 : g.drawLine(x1,y1,x2,y2); break;
        case 1 : g.fillRect(x1,y1,50,50); break;
        case 2 : g.fillOval(x1,y1,diameter,diameter);
    }
}
```

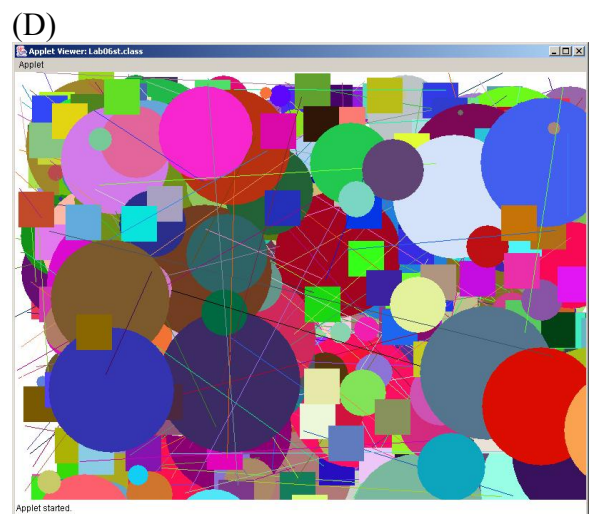
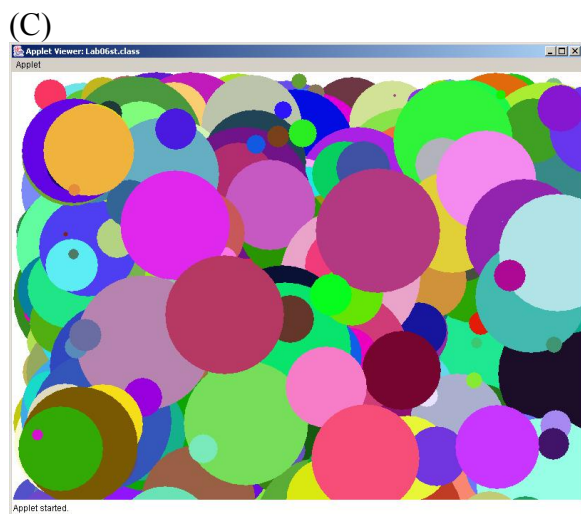
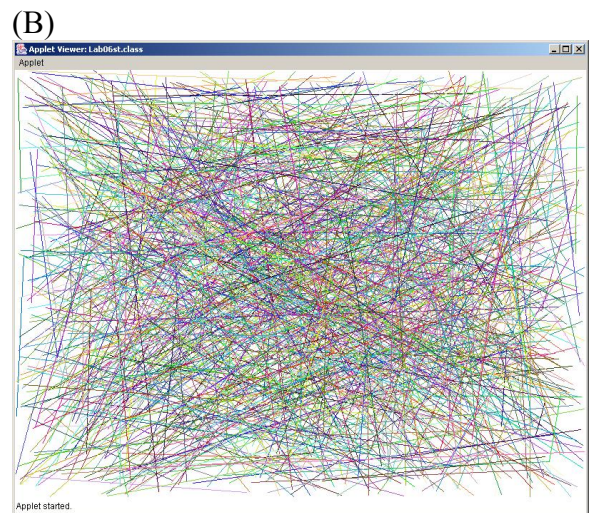


(E) No Output

62. What is the output of this program segment?

```
Random rnd = new Random(1234);
for (int count = 1; count <= 1000; count++)
{
    int red = rnd.nextInt(256);
    int green = rnd.nextInt(256);
    int blue = rnd.nextInt(256);
    g.setColor(new Color(red, green, blue));
    int x1 = rnd.nextInt(800);
    int y1 = rnd.nextInt(600);
    int x2 = rnd.nextInt(800);
    int y2 = rnd.nextInt(600);
    int diameter = rnd.nextInt(200);
    int shape = 1;

    switch (shape)
    {
        case 0 : g.drawLine(x1,y1,x2,y2); break;
        case 1 : g.fillRect(x1,y1,50,50); break;
        case 2 : g.fillOval(x1,y1,diameter,diameter);
    }
}
```

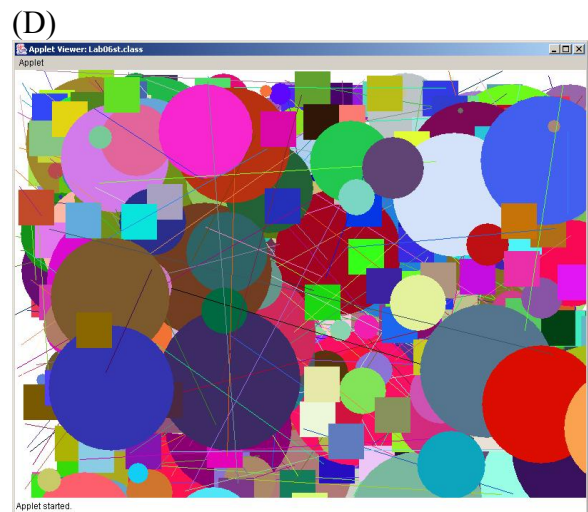
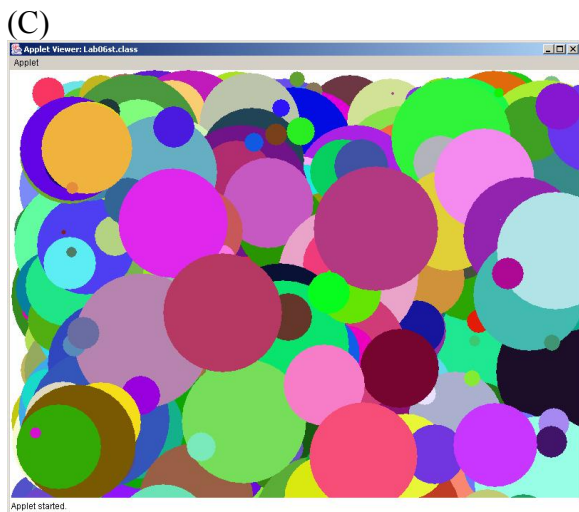
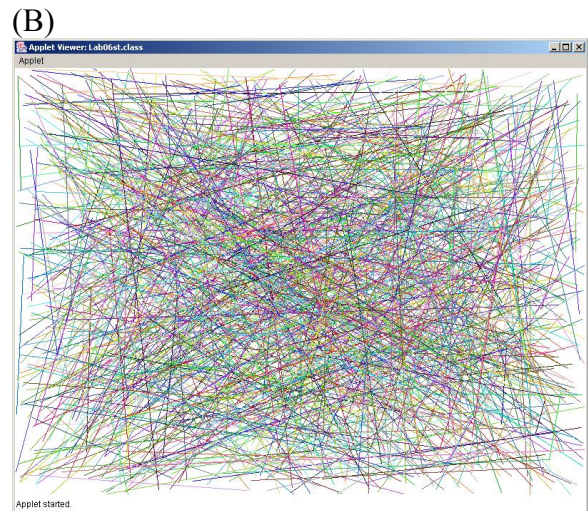


(E) No Output

63. What is the output of this program segment?

```
Random rnd = new Random(1234);
for (int count = 1; count <= 1000; count++)
{
    int red = rnd.nextInt(256);
    int green = rnd.nextInt(256);
    int blue = rnd.nextInt(256);
    g.setColor(new Color(red, green, blue));
    int x1 = rnd.nextInt(800);
    int y1 = rnd.nextInt(600);
    int x2 = rnd.nextInt(800);
    int y2 = rnd.nextInt(600);
    int diameter = rnd.nextInt(200);
    int shape = 2;

    switch (shape)
    {
        case 0 : g.drawLine(x1,y1,x2,y2); break;
        case 1 : g.fillRect(x1,y1,50,50); break;
        case 2 : g.fillOval(x1,y1,diameter,diameter);
    }
}
```



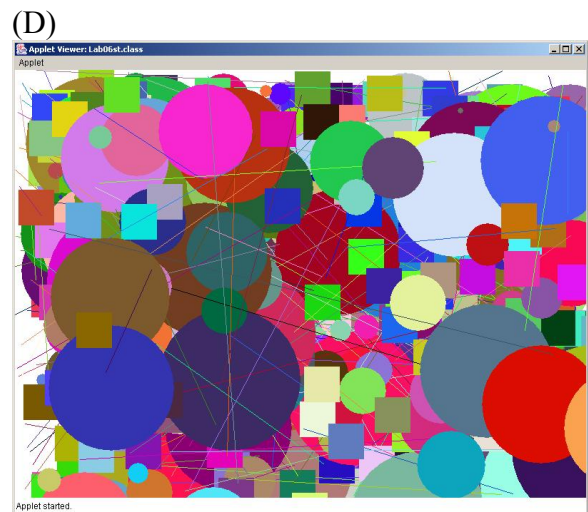
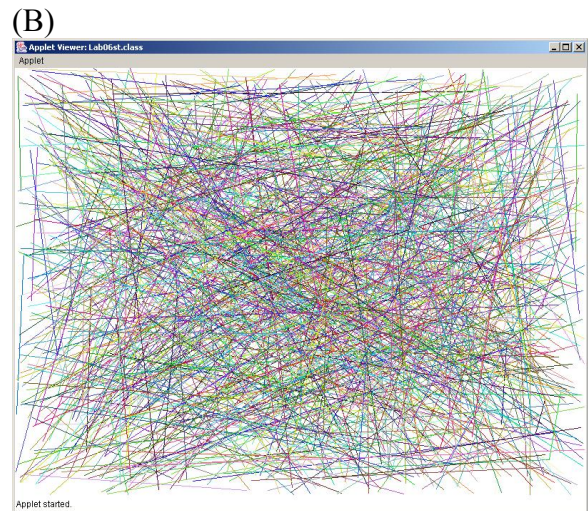
(E) No Output



64. What is the output of this program segment?

```
Random rnd = new Random(1234);
for (int count = 1; count <= 1000; count++)
{
    int red = rnd.nextInt(256);
    int green = rnd.nextInt(256);
    int blue = rnd.nextInt(256);
    g.setColor(new Color(red, green, blue));
    int x1 = rnd.nextInt(800);
    int y1 = rnd.nextInt(600);
    int x2 = rnd.nextInt(800);
    int y2 = rnd.nextInt(600);
    int diameter = rnd.nextInt(200);
    int shape = 3;

    switch (shape)
    {
        case 0 : g.drawLine(x1, y1, x2, y2); break;
        case 1 : g.fillRect(x1, y1, 50, 50); break;
        case 2 : g.fillOval(x1, y1, diameter, diameter);
    }
}
```

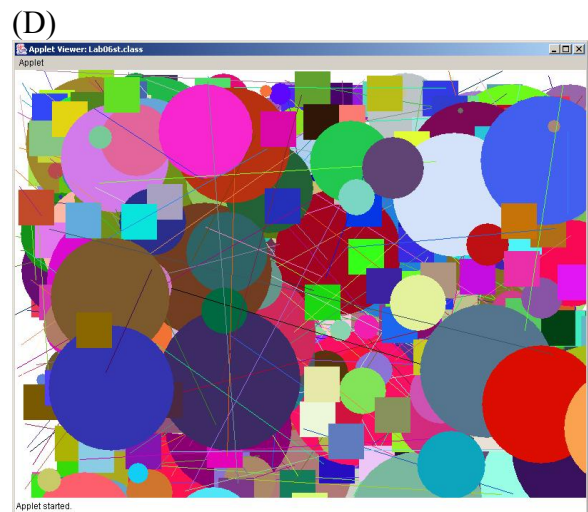
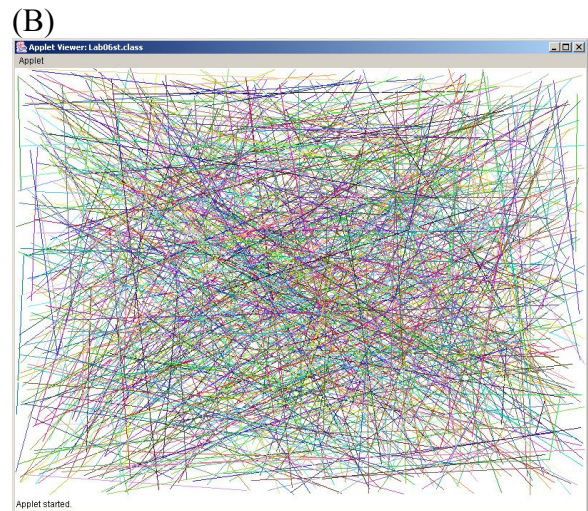


(E) No Output

65. What is the output of this program segment?

```
Random rnd = new Random(1234);
for (int count = 1; count <= 1000; count++)
{
    int red = rnd.nextInt(256);
    int green = rnd.nextInt(256);
    int blue = rnd.nextInt(256);
    g.setColor(new Color(red, green, blue));
    int x1 = rnd.nextInt(800);
    int y1 = rnd.nextInt(600);
    int x2 = rnd.nextInt(800);
    int y2 = rnd.nextInt(600);
    int diameter = rnd.nextInt(200);
    int shape = rnd.nextInt(3);

    switch (shape)
    {
        case 0 : g.drawLine(x1,y1,x2,y2); break;
        case 1 : g.fillRect(x1,y1,50,50); break;
        case 2 : g.fillOval(x1,y1,diameter,diameter);
    }
}
```



(E) No Output