Practice Exam #5

SECTION I

Time — 1 hour and 30 minutes
Number of questions — 40
Percent of total grade — 50

1. Consider the following code segment.

```
double x = 5*4/2 - 5/2*4;
System.out.println(x);
```

What is printed when the code segment is executed?

(A) 0
(B) 1
(C) 0.0
(D) 1.0
(E) 2.0

2. Consider the following method and code segment in the same class.

```
public void process(String s)
{
  s = s.substring(2, 3) + s.substring(1, 2) +
                                    s.substring(0, 1);
}

  String s = "ABCD";
  process(s);
  System.out.println(s);
```

What is printed when the code segment is executed?

(A) ABCD
(B) CBA
(C) CDBCA
(D) CDBCAB
(E) StringIndexOutOfBoundsException

3. Assume that `int` variables `x` and `y` have been properly declared and initialized. The expression

```
!(x > y && y <= 0)
```

is equivalent to which of the following?

(A) `x <= y || y > 0`
(B) `x > y && y <= 0`
(C) `x > y || y < 0`
(D) `x <= y && y <= 0`
(B) `!(x <= y) || (y > 0)`

4. Consider the following method.

```
public int guess(int num1, int num2)
{
  if (num1 % num2 == 0)
    return num2;
  else
    return guess(num2, num1 % num2);
}
```

What is the value of `num` after the following statement is executed?

```
int num = (6 * 14) / guess(6, 14);
```

(A) 6
(B) 12
(C) 28
(D) 42
(E) 84

5. Consider the following code segment.

```
ArrayList<String> words = new ArrayList<String>();
words.set(0, "ANT");
```

What is the result when the code segment is executed?

(A) `NullPointerException`
(B) `IndexOutOfBoundsException`
(C) The size of `words` remains 0 and nothing is added to `words`
(D) The size of `words` becomes 1 and the element at index 0 becomes `"ANT"`
(E) The size of `words` becomes 10 and the element at index 0 becomes `"ANT"`

6. Assume that `String` variables `str1` and `str2` have been properly declared and initialized. Which of the following `boolean` expressions evaluate to `true` if and only if `str1` and `str2` hold the same values?

I. `str1 == str2`

II. `str1.equals(str2)`

III. `str1.compareTo(str2) == 0`

(A) I only
(B) II only
(C) I and II only
(D) II and III only
(E) I, II, and III

7. Consider the following code segment.

```
int[] nums = new int[51];

for (int k = 0; k < nums.length; k++)
  nums[k] = 1;

for (int k = 3; k <= 50; k += 3)
  nums[k] = 0;

for (int k = 5; k <= 50; k += 5)
  nums[k] = 0;
```

How many elements in the array `nums` have the value 0 after this code has been executed?

(A) 23
(B) 25
(C) 26
(D) 27
(E) 28

8. Which of the following methods are equivalent (always return the same value for the same values of input parameters)?

I.
```
public boolean fun(int a, int b, int c)
{
  if (a >= b)
    if (b >= c)
      return true;
    else
      return false;
  else
    return false;
}
```

II.
```
public boolean fun(int a, int b, int c)
{
  if (a >= b && b >= c)
    return true;
  else
    return false;
}
```

III.
```
public boolean fun(int a, int b, int c)
{
  return a >= b || b >= c;
}
```

(A) I and II only
(B) I and III only
(C) II and III only
(D) All three are equivalent
(E) All three are different

9. Consider the following method.

```
/** Precondition: amt represents a positive value in dollars
 *                and cents (for example, 1.15 represents
 *                one dollar and fifteen cents).
 */
private int process(double amt)
{
  return (int)(amt * 100 + 0.5) % 100;
}
```

Which of the following best describes its return value?

(A) The cent portion in `amt`
(B) The number of whole dollars in `amt`
(C) `amt` converted into cents
(D) `amt` rounded to the nearest dollar
(E) The smallest number of whole dollars that is greater than or equal to `amt`

10. Consider the following method with two missing statements.

```
/** Returns the sum of all positive odd values
 *  among the first n elements of arr.
 *  Precondition: 1 <= n <= arr.length.
 */
public static int addPositiveOddValues(int[] arr, int n)
{
  int sum = 0;

  < statement 1 >
  {
    < statement 2 >
      sum += arr[i];
  }
  return sum;
}
```

Which of the following are appropriate replacements for < *statement 1* > and < *statement 2* > so that the method works as specified?

	< *statement 1* >	< *statement 2* >
(A)	`for (int i = 1; i < n; i += 2)`	`if (arr[i] > 0)`
(B)	`for (int i = 0; i < n; i++)`	`if (arr[i] > 0 && arr[i] % 2 != 0)`
(C)	`for (int i = 1; i <= n; i += 2)`	`if (arr[i] > 0)`
(D)	`for (int i = 0; i <= n; i++)`	`if (arr[i] % 2 != 0)`
(E)	None of the above	

11. Consider the following code segment.

```
int n = IO.readInt();    // read an int value
n = Math.abs(n);

while (n >= 2)
{
  n = n/2 - 1;
}
System.out.println(n);
```

Which of the following represents the list of all possible outputs?

(A) `0`
(B) `-1, 0`
(C) `0, 1`
(D) `-1, 1`
(E) `-1, 0, 1`

12. Consider the following code segment.

```
int x = (int)(2*Math.random()) + (int)(2*Math.random());
System.out.println(x);
```

Which of the following could be printed by the code segment?

(A) 0 only
(B) 2 only
(C) 0 and 2 only
(D) 1 and 2 only
(E) 0, 1, and 2

13. Esther wants to write a method `swap` that swaps two integer values. Which of the following three ways of representing the values and corresponding methods successfully swap the values?

I.
```
/** a and b are Integer objects that represent
 *  the values to be swapped
 */
public static void swap(Integer a, Integer b)
{
  Integer temp = a; a = b; b = temp;
}
```

II.
```
/** a[0] and a[1] contain the values to be swapped */
public static void swap(int[] a)
{
  int temp = a[0]; a[0] = a[1]; a[1] = temp;
}
```

III.
```
/** a[0] and b[0] contain the values to be swapped */
public static void swap(int[] a, int[] b)
{
  int temp = a[0]; a[0] = b[0]; b[0] = temp;
}
```

(A) I only
(B) II only
(C) I and II only
(D) II and III only
(E) I, II, and III

14. Consider the following method with missing code.

```
public void zeroSomething(int[][] m)
{
  int numRows = m.length;
  int numCols = m[0].length;

  < missing code >
}
```

Which of the following three versions of < *missing code* > are equivalent, that is, result in the same values for a given two-dimensional array `m`?

I.
```
for (int k = 0; k < numRows; k++)
{
  m[k][0] = 0;
  m[k][numCols - 1] = 0;
}
for (int k = 0; k < numCols; k++)
{
  m[0][k] = 0;
  m[numRows - 1][k] = 0;
}
```

II.
```
for (int k = 0; k < numRows - 1; k++)
{
  m[k][0] = 0;
  m[numRows - k - 1][numCols - 1] = 0;
}
for (int k = 0; k < numCols - 1; k++)
{
  m[0][numCols - k - 1] = 0;
  m[numRows - 1][k] = 0;
}
```

III.
```
for (int k = 0; k < numCols; k++)
{
  m[0][k] = 0;
  m[numRows - 1][k] = 0;
}
for (int[] r : m)
{
  r[0] = 0;
  r[numCols - 1] = 0;
}
```

(A) All three are equivalent
(B) I and II only
(C) II and III only
(D) I and III only
(E) All three are different

15. Consider the following method, which prints a message depending on whether the equation $x^2 + bx + c = 0$ has a positive solution.

```
public static void positiveSolution(double b, double c)
{
  double d2 = b*b - 4*c;

  < missing code >
}
```

Which of the following replacements for < *missing code* > are equivalent to each other, that is, produce the same outcome given any values of parameters `b` and `c`?

I.
```
double x = (-b + Math.sqrt(d2)) / 2;
if (d2 >= 0 && x > 0)
  System.out.println("Has a positive solution");
else
  System.out.println("No positive solutions");
```

II.
```
if (d2 >= 0)
{
  double x = (-b + Math.sqrt(d2)) / 2;
  if (x > 0)
    System.out.println("Has a positive solution");
}
else
  System.out.println("No positive solutions");
```

III.
```
double x = -1.0;
if (d2 >= 0)
  x = (-b + Math.sqrt(d2)) / 2;
if (x > 0)
  System.out.println("Has a positive solution");
else
  System.out.println("No positive solutions");
```

(A) I and II only
(B) I and III only
(C) II and III only
(D) All three are equivalent
(E) All three are different

16. Consider the following class definitions.

```
public class Airplane
{
  private int fuel;

  public Airplane() { fuel = 0; }
  public Airplane(int g) { fuel = g; }

  public void addFuel() { fuel++; }
  public String toString() { return fuel + " "; }
}

public class Jet extends Airplane
{
  public Jet(int g) { super(2*g); }
}
```

What is the result when the following code is compiled/executed?

```
Airplane plane = new Airplane(4);
Airplane jet = new Jet(4);

System.out.print(plane);
plane.addFuel();
System.out.print(plane);

System.out.print(jet);
jet.addFuel();
System.out.print(jet);
```

(A) A syntax error, "undefined addFuel," is reported for the `jet.addFuel();` statement.
(B) The code compiles and runs with no errors; the output is `4 5 4 5`
(C) The code compiles and runs with no errors; the output is `4 5 5 6`
(D) The code compiles and runs with no errors; the output is `4 5 8 9`
(E) The code compiles and runs with no errors; the output is `8 9 9 10`

17. What is printed when the following statement is executed?

```
int x = 1024;
System.out.println((1 - 2*x) / x);
```

(A) `-2`
(B) `-1`
(C) `0`
(D) `1`
(E) `2`

Questions 18 and 19 refer to the following method `process`.

```
public void process(int[] a)
{
  for (int i = 1; i < a.length; i++) // Line 1
  {
    int current = a[i];              // Line 2
    int j = 0;                       // Line 3

    while (a[j] < current)           // Line 4
    {
      j++;                           // Line 5
    }

    for (int k = i; k > j; k--)      // Line 6
    {
      a[k] = a[k-1];                 // Line 7
    }

    a[j] = current;                  // Line 8
  }
}
```

18. The algorithm implemented in the method `process` is best described as:

(A) Mergesort
(B) Insertion Sort
(C) Selection Sort
(D) A faulty implementation of Binary Search
(E) A faulty implementation of a sorting algorithm

19. Given

```
int[] a = {24, 16, 68, 56, 32};
```

what will be the result after the statement on Line 8 in `process` completes for the second time?

(A) The values in `a` are 16, 24, 68, 56, 32
(B) The values in `a` are 16, 24, 32, 56, 68
(C) The values in `a` are 24, 16, 32, 56, 68
(D) The code has failed with an `ArrayIndexOutOfBoundsException` on Line 4
(E) The code has failed with an `ArrayIndexOutOfBoundsException` on Line 8

20. The two versions of the `search` method shown below are both intended to return `true` if `ArrayList<Object> list` contains the target value; otherwise they are supposed to return `false`.

Version 1

```
public boolean search(ArrayList<Object> list, Object target)
{
  for (Object x : list)
  {
    if (target.equals(x))
      return true;
  }
  return false;
}
```

Version 2

```
public boolean search(ArrayList<Object> list, Object target)
{
  boolean found = false;

  for (Object x : list)
  {
    if (target.equals(x))
      found = true;
  }
  return found;
}
```

Which of the following statements about the two versions of `search` is true?

(A) Only Version 1 works as intended.
(B) Only Version 2 works as intended.
(C) Both versions work as intended; Version 1 is often more efficient than Version 2.
(D) Both versions work as intended; Version 2 is often more efficient than Version 1.
(E) Both versions work as intended; the two versions are always equally efficient.

21. What is the best description of the return value of the method `propertyX` below?

```
/** Precondition: v.length >= 2
 */
public boolean propertyX(int[] v)
{
  boolean flag = false;

  for (int i = 0; i < v.length - 1; i++)
  {
    flag = flag || (v[i] == v[i+1]);
  }

  return flag;
}
```

(A) Returns `true` if the elements of `v` are sorted in ascending order, `false` otherwise
(B) Returns `true` if the elements of `v` are sorted in descending order, `false` otherwise
(C) Returns `true` if `v` has two adjacent elements with the same value, `false` otherwise
(D) Returns `true` if `v` has two elements with the same value, `false` otherwise
(E) Returns `true` if all elements in `v` have different values, `false` otherwise

22. Consider the following method.

```
/** Returns the greatest common divisor of a and b.
 *  Precondition: a > 0 and b > 0
 */
public static int gcd(int a, int b)
{
  int d = a;
  if (b < d)
    d = b;

  while (a % d != 0 || b % d != 0)
    d--;

  return d;
}
```

How many times will modulo division be performed when `gcd(12, 15)` is called?

(A) 14
(B) 20
(C) 24
(D) 27
(E) 30

23. Consider the following recursive method.

```
public static void printSomething(int n)
{
  if (n > 0)
    printSomething(n-1);

  for (int k=1; k <= n; k++)
    System.out.print(k);
}
```

What is printed when `printSomething(4)` is called?

(A) `1234`
(B) `1231234`
(C) `1121231234`
(D) `1234123121`
(E) `4444333221`

24. Suppose the class `Text` has a constructor that takes one parameter of `String` type. `Text` also has a static method `translate`:

```
public static Text translate(Text t) { /* code not shown */ }
```

The class `Greeting` extends `Text` and has a constructor that takes one parameter of `String` type. Which of the following code segments in a client class of `Text` and `Greeting` will compile with no errors?

I.
```
Text greeting = new Text("Hello");
Text translation = Text.translate(greeting);
```

II.
```
Text greeting = new Greeting("Hello");
Text translation = Text.translate(greeting);
```

III.
```
Greeting greeting = new Greeting("Hello");
Text translation = Text.translate(greeting);
```

(A) I only
(B) I and II only
(C) I and III only
(D) II and III only
(E) I, II, and III

25. Consider the following method.

```
/** Returns the index of searchVal, if found in list;
 *  otherwise returns -1
 */
public int binarySearch(ArrayList<String> list,
                                    String searchVal)
{
  int first = 0, last = list.size() - 1;

  while (first <= last)
  {
    int mid = (first + last) / 2;

    if (searchVal.compareTo(list.get(mid)) < 0)
      last = mid - 1;
    else if (searchVal.compareTo(list.get(mid)) > 0)
      first = mid + 1;
    else
      return mid;    // Statement 1
  }
  return -1;         // Statement 2
}
```

We want to modify this method: if `searchVal` is not already in the list, we want `binarySearch` to return the position where it can be inserted to keep the list sorted. Which of the following could be used for *Statement 1* and *Statement 2*?

	Statement 1	*Statement 2*
(A)	`return mid;`	`return mid;`
(B)	`return mid;`	`return first;`
(C)	`return mid;`	`return last;`
(D)	`return mid - 1;`	`return first;`
(E)	`return mid - 1;`	`return last;`

26. The statement

```
System.out.println(Integer.MAX_VALUE);
```

prints `2147483647`. What does the following statement print?

```
System.out.println(Integer.MAX_VALUE / 10 + 1);
```

(A) Nothing: it causes a syntax error
(B) `214748365.7`
(C) `214748365.0`
(D) `214748365`
(E) Nothing: it causes an `ArithmeticException`

27. Consider the following code segment.

```
int[][] t = new int[2][3];

for (int i = 0; i < t.length; i++)
{
  for (int j = 0; j < t[0].length; j++)
  {
    t[i][j] = i + j + 1;
  }
}
```

What is the result when the code segment is executed?

(A) `t` holds the values

```
1 2 3
4 5 6
```

(B) `t` holds the values

```
1 2 0
2 3 0
```

(C) `t` holds the values

```
1 2 3
2 3 4
```

(D) `t` holds the values

```
3 4 5
4 5 6
```

(E) `ArrayIndexOutOfBoundsException`

28. Consider the following two recursive versions of the method `choose(n, k)`.

Version 1

```
public static int choose(int n, int k)
{
  if (k == 0)
    return 1;
  else
    return choose(n, k-1) * (n-k+1)/k;
}
```

Version 2

```
public static int choose(int n, int k)
{
  if (k < 0 || k > n)
    return 0;
  else if (n == 0)
    return 1;
  else
    return choose(n-1, k-1) + choose(n-1, k);
}
```

When `choose(4, 2)` is called, how many times total, including the original call, will `choose` be called in each version?

	Version 1	Version 2
(A)	2	7
(B)	2	19
(C)	3	7
(D)	3	19
(E)	3	27

29. What is the outcome when the following code is compiled/executed?

```
ArrayList<Integer> numbers = new ArrayList<Integer>();
numbers.add(1);
numbers.add(2);
numbers.add(3);
numbers.add(4);

for (int x : numbers)
{
  if (x % 2 == 0)
    numbers.add(5);
}
System.out.println(numbers);
```

(A) Syntax error
(B) `IndexOutOfBoundsException`
(C) `ConcurrentModificationException`
(D) `[1, 2, 3, 4, 5, 5]` is printed
(E) `[5, 5, 1, 2, 3, 4]` is printed

30. Consider the following class.

```
public class Question
{
  private boolean answer;

  public void flip(Question q)
  {
    < missing statement >
  }

  /* constructors and other methods not shown */

}
```

Which of the following could replace < *missing statement* > in the `flip` method so that it compiles with no errors?

I. `answer = !answer;`
II. `answer = !q.answer;`
III. `q.answer = !q.answer;`

(A) None
(B) I only
(C) II only
(D) I and II only
(E) All three

Questions 31-34 use the classes `Track` and `CD`:

```
public class Track
{
  private String name;
  private int duration;

  public Track(String nm, int dur)
  {
    name = nm;
    duration = dur;
  }

  public String getName() { return name; }
  public int getDuration() { return duration; }
}

public class CD
{
  private String title;
  private String band;
  private int numTracks;
  private ArrayList<Track> tracks;

  public CD(String t, int n)
  {
    title = t;
    numTracks = n;
  }

  /** Initializes all the instance variables and copies
   *  all the data from songs into tracks.
   */
  public CD(String t, String b, int n, ArrayList<Track> songs)
  {
    title = t;
    band = b;
    numTracks = n;

    < missing code >
  }

  public int totalPlayTime()
  { /* implementation not shown */ }

  /** Returns duration of the k-th track.
   *  Precondition: 1 <= k <= numTracks.
   */
  public int getDuration(int k)
  { /* implementation not shown */ }

  /* other methods not shown */
}
```

31. Which of the following declarations is INVALID?

(A) `Track tune = new Track();`
(B) `Track tune = new Track("Help", 305);`
(C) `Track[] playList = new Track[20];`
(D) `CD top = new CD("throwing copper", 13);`
(E) `CD[][] rack = new CD[3][40];`

32. Which of the following expressions correctly refers to the duration of the *k*-th track inside `CD`'s method `totalPlayTime`?

(A) `getDuration(k);`
(B) `tracks[k-1].duration;`
(C) `tracks.getDuration(k);`
(D) `tracks.get(k-1).duration;`
(E) `getDuration(tracks[k-1]);`

33. Consider the following code segment.

```
Track t = new Track("lightning crashes", 200);
ArrayList<Track> tracks = new ArrayList<Track>();  // Line **

for (int count = 1; count <= 13; count++)
{
  tracks.add(t);
}

CD live = new CD("throwing copper", "live", 13, tracks);
System.out.println(live.totalPlayTime());
```

What is the result when the code segment is compiled/executed?

(A) Syntax error on Line **
(B) Run-time `IndexOutOfBoundsException`
(C) 0 is printed
(D) 200 is printed
(E) 2600 is printed

34. Which of the following can replace < *missing code* > in the `CD` class's constructor so that it works as specified?

I.
```
for (Track t : songs)
  tracks.add(t);
```

II.
```
tracks = new ArrayList<Track>();
for (Track t : songs)
  tracks.add(t);
```

III.
```
tracks = new ArrayList<Track>();
for (int i = 0; i < songs.size(); i++)
  tracks.set(i, songs.get(i));
```

(A) I only
(B) II only
(C) I and II only
(D) II and III only
(E) I, II, and III

35. Consider the following class.

```
public class Counter
{
  private int count = 0;

  public Counter() { count = 0; }
  public Counter(int x) { count = x; }              // Line 1
  public int getCount() { return count; }           // Line 2
  public void setCount(int c) { int count = c; }    // Line 3
  public void increment() { count++; }              // Line 4
  public String toString() { return "" + count; } // Line 5
}
```

The test code

```
Counter c = new Counter();
c.setCount(3);
c.increment();
System.out.println(c.getCount());
```

is supposed to print `4`, but the class has an error. What is actually printed, and which line in the class definition should be changed to get the correct output, `4`?

(A) 0, Line 1
(B) 1, Line 3
(C) 0, Line 4
(D) 3, Line 4
(E) 36, Line 5

36. Consider the following class.

```
public class Matrix
{
  private int[][] m;

  /** Initializes m to a square n-by-n array with all
   *  the elements on the diagonal m[k][k] equal to 0 and
   *  all other elements equal to 1
   */
  public Matrix(int n)
  {
    m = new int[n][n];

    < missing code >
  }

  < other constructors and methods not shown >
}
```

Which of the following could replace < *missing code* > in `Matrix`'s constructor so that it compiles with no errors and works as specified?

I.
```
for (int r = 0; r < n; r++)
  for (int c = 0; c < n; c++)
    m[r][c] = 1;
for (int k = 0; k < n; k++)
  m[k][k] = 0;
```

II.
```
for (int r = 0; r < n; r++)
  for (int c = 0; c < n; c++)
    if (r != c)
      m[r][c] = 1;
```

III.
```
for (int c = 0; c < n; c++)
  for (int r = 0; r < n; r++)
    if (r == c)
       m[r][c] = 0;
    else
       m[r][c] = 1;
```

(A) I only
(B) II only
(C) I and II only
(D) II and III only
(E) I, II, and III

Questions 37 and 38 are based on the following information.

Ashanti has decided to write and test her own version of the `indexOf` method, which takes two strings as parameters:

```
/** Returns the first index k such that word.substring(k)
 *  starts with str; if no such k found, returns -1.
 *  Precondition: word != null, str != null.
 */
public int indexOf(String word, String str)
{
  int n = str.length();
  for (int k = 0; k <= word.length() - n; k++)
    if (word.substring(k, k + n).equals(str))
      return k;
  return -1;
}
```

37. What will happen if Morgan calls Ashanti's `indexOf` method as follows?

```
int i = indexOf("Hello", null);
```

(A) `i` will be set to 0
(B) `i` will be set to -1
(C) `NullPointerException`
(D) `StringIndexOutOfBoundsException`
(E) `ConcurrentModificationException`

38. Ashanti also wrote the test method `findWord` —

```
public String findWord(ArrayList<String> words, String str)
{
  for (String word : words)
    if (indexOf(word, str) >= 0)
      return word;
  return null;
}
```

— and created a list of strings `continents` with the values

```
["Africa", "America", "Antarctica", "Asia", "Australia",
"Europe"]
```

How many times will the method `equals` be called in Ashanti's `indexOf` code when `findWord(continents, "tar")` is called?

(A) 1
(B) 3
(C) 12
(D) 15
(E) 36

39. Consider the following code segment.

```
ArrayList<Integer> list = new ArrayList<Integer>();
list.add(1);
list.add(2);

for (int i = 1; i <= 3; i++)
{
  list.add(i, i);
}

System.out.println(list);
```

What is the result when the code segment is executed?

(A) `[1, 1, 2, 2, 3]` is printed
(B) `[1, 1, 2, 3, 2]` is printed
(C) `[1, 2, 1, 2, 3]` is printed
(D) `[1, 2, 3, 1, 2]` is printed
(E) `IndexOutOfBoundsException`

40. Consider the following method.

```
public void reduce(int[] arr, int len)
{
  for (int k = 0; k < len; k++)
  {
    arr[k]--;
  }

  len--;
}
```

What is printed when the following code segment is executed?

```
int[] counts = {3, 2, 1, 0};
int len = 3;
reduce(counts, len);

for (int c : counts)
{
  System.out.print(c + " ");
}

System.out.println(len);
```

(A) 2 1 0 -1 2
(B) 2 1 0 -1 3
(C) 2 1 0 0 2
(D) 2 1 0 0 3
(E) 2 1 1 0 3

Practice Exam #5

SECTION II

Time — 1 hour and 30 minutes
Number of questions — 4
Percent of total grade — 50

1. A high school Casino Night event offers a game of chance in which a player can buy a ticket to have a low-probability chance at winning a fixed prize. (All tickets and prizes are in the fictional *Darsek* currency). The class `GameOfChance` models this game. You will write two methods of this class.

```
public class GameOfChance
{
  /** Returns true if a game is won; otherwise returns false.
   */
  private static boolean win()
  { /* implementation not shown */ }

  /** Returns the average amount won or lost
   *  after playing n games, given the price of one ticket
   *  and the size of the prize.
   */
  public static double averageAmt(int prize,
                                  double ticketPrice, int n)
  { /* to be implemented in part (a) */ }

  /** Returns the prize size (a multiple of 5) that makes
   *  the game worth playing, based on the average return
   *  after 100 games, as modeled by the averageAmt method.
   */
  public static int prizeWorthPlaying(double ticketPrice)
  { /* to be implemented in part (b) */ }
}
```

(a) Write the `averageAmt` method. The method takes three parameters: the size of the prize, the ticket price, and the number of games played. `averageAmt` calls the `win` method for each game played to determine whether a game was won or lost: `win` returns `true` to indicate that a game was won and `false` if a game was lost. The price of one ticket is subtracted from the total for each game played, regardless of whether the game was won or lost. For example, `averageAmt(15, 1.0, 10)` will call `win()` ten times. If none of these calls return `true`, then `averageAmt` will return $(0-10\cdot 1.0)/10=-1.0$; if, however, exactly one of the ten calls to `win` returns `true`, then `averageAmt` will return $(15-10\cdot 1.0)/10=0.5$.

Complete the `averageAmt` method below.

```
/** Returns the prize size (a multiple of 5) that makes
 *  the game worth playing, based on the average return
 *  after 100 games, as modeled by the averageAmt method.
 */
public static double averageAmt(int prize,
                                double ticketPrice, int n)
```

(b) Write the `prizeWorthPlaying` method of the `GameOfChance` class. This method tries different values of `prize`, in increments of 5 — 5, 10, 15, and so on — until the size of the prize makes it worth playing the game, given the price of one ticket. `prizeWorthPlaying` relies on the result of `averageAmt` for 100 games: it needs to be positive to make the game worth playing. For example, if `averageAmt(15, 1.7, 100)` returns -0.2 and `averageAmt(20, 1.7, 100)` returns 1.1, then `prizeWorthPlaying(1.7)` will return 20.

Complete the `prizeWorthPlaying` method below.

```
/** Returns the prize size (a multiple of 5) that makes
 *  it worth playing, based on the average return after
 *  100 games, as modeled by the averageAmt method.
 */
public static int prizeWorthPlaying(double ticketPrice)
```

For additional practice, implement the `win` method. Make it return `true` randomly with the probability 0.05. See if the value returned by `prizeWorthPlaying` makes sense with your `win` method and a ticket price of $1.00 — it should usually be between 15 and 25.

2. You can purchase any number of apples at Papas Orchards, but every six purchased are priced as five. For example, if the price of each apple is 69 cents and you purchase 34 apples, your total cost will be 29 * $0.69 = $20.01, because 30 apples are priced as 25. The class `ApplesPurchase` helps calculate the cost of your purchase at Papas.

   The constructor of the `ApplesPurchase` class takes two `int` parameters: the price of one apple in cents and the number of apples purchased. `ApplesPurchase` has three methods:

   - `totalCost` — takes no parameters; returns the total cost of the purchase in cents.
   - `toDollarsAndCents` — returns a `String` that represents a given `int` parameter `cents` as dollars and cents, with a dollar sign, at least one digit for dollars, a decimal point, and two digits after the decimal point.
   - `toString` — returns a string with the summary of the purchase (in the same format as the one shown in the example below).

   For example,

```
ApplesPurchase purchase1 = new ApplesPurchase(79, 1);
System.out.println(purchase1);
ApplesPurchase purchase2 = new ApplesPurchase(50, 12);
System.out.println(purchase2);
ApplesPurchase purchase3 = new ApplesPurchase(69, 34);
System.out.println(purchase3);
```

   in a client class of `ApplesPurchase` should print

```
1 apples at $0.79 each: $0.79
12 apples at $0.50 each: $5.00
34 apples at $0.69 each: $20.01
```

   Do not duplicate code from `totalCost` and `toDollarsAndCents` in your `toString` method; call these methods instead.

   Write a complete `ApplesPurchase` class, including the required instance variables and the constructor and three methods described above. Your implementation must meet the specifications, and the format of the output must conform to the above example.

   For additional practice, derive a class `ApplesPurchaseWithPie` from `ApplesPurchase` to provide a way to add an apple pie to the purchase. Apple pies should be available in two sizes, small (`"S"`) and large (`"L"`). Pass the price of the large pie as a parameter to `ApplesPurchaseWithPie`'s constructor. Compute the price of the small pie as 75% of the large pie price, rounded to the nearest cent. Override `ApplesPurchase`'s `totalCost` and `toString` methods.

3. In an international ballroom dancing competition, competitors are scored by a panel of judges on a scale from 0 to 100. The final score for each couple is computed as the average of the scores from all the judges (possibly excluding the highest score and/or the lowest score in some cases, as explained in Part (b)).

   (a) Write a method `findMaxAndMin` of the `BallroomScoring` class. This method takes an `ArrayList<Integer> scores` as a parameter and returns an array of length two, where the first element is the maximum value in `scores` and the second element is the minimum value in `scores`.

   Complete the method `findMaxAndMin` below.

```
/** Returns an array of two elements in which
 *  the first element is the maximum value in scores and
 *  the second element is the minimum value in scores.
 *  Precondition: scores.size() > 0; 0 <= all scores <= 100
 */
public static int[] findMaxAndMin(ArrayList<Integer> scores)
```

   (b) Write a method `averageScore` of the `BallroomScoring` class that takes an `ArrayList<Integer> scores` as a parameter and computes and returns the average score. However, if the size of the list `scores` is 6 or greater and the maximum value occurs only once in the array, that value is excluded from computing the average. The same is done for the minimum value.

   The `BallroomScoring` class has a private method

```
private static int countOccurrences(ArrayList<Integer> scores,
                                    int target)
```

   that returns the number of occurrences of `target` in `scores`. In writing `averageScore`, call the `countOccurrences` method. Also assume that the method `findMaxAndMin` from Part (a) works as specified, regardless of what you wrote there. You may not receive full credit if instead of calling these two methods you re-implement their functionality here.

   Complete the method `averageScore` below.

```
/** Returns the average of the values in scores. However,
 *  if the size of the scores list is not less than 6 and
 *  the largest value occurs only once in scores, that value
 *  is excluded from computing the average; the same for
 *  the smallest value.
 *  Precondition: scores.size() >= 3
 */
public static double averageScore(ArrayList<Integer> scores)
```

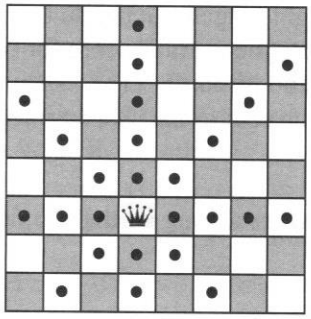
For additional practice, implement the `countOccurrences` method.

Then implement a computer model for a ballroom dancing competition. The competition includes several dance events, and several couples participate in all the events. Each event is scored by a panel of eight judges. Generate the judges' scores as random integers from 90 to 99. Calculate the average score for each couple in each event by calling the `averageScore` method from Part (b). Then add up the couple's average scores in all events and round the sum to the nearest integer to get the couple's final score.

Determine the maximum total score among all the couples, and find the number of couples that share first place.

Whenever possible, use the appropriate methods of the `BallroomScoring` class — do not duplicate their code.

4. In chess, a queen can move vertically, horizontally, and diagonally on an 8-by-8 chessboard. The picture below shows all the positions on the board that a queen can reach from its current position.



Squares on the chessboard are often labeled with a letter and a digit, but we will just treat a chessboard as a two-dimensional array and use indices, as in Java, counting from zero. So the upper-left corner of the board is in row 0, column 0, and the position of the queen in the above picture is row 5, column 3.

(a) Write a method `oneQueen` that takes two integer parameters, the row and column of a queen on a chessboard, and generates and returns an 8-by-8 two-dimensional array of `boolean` values in which the queen's position and all the positions that the queen can reach from it are set to `true`, and all the other positions are set to `false`.

Complete the method `oneQueen` below.

```
/** Returns an 8-by-8 boolean two-dimensional array in which
 *  the element at row, col and all the elements that
 *  represent positions reachable by a queen placed at row,
 *  col are set to true, and all the other elements are set
 *  to false.
 *  Precondition: 0 <= row < 8; 0 <= col < 8
 */
public static boolean[][] oneQueen(int row, int col)
```

(b) Write a method `twoQueens` that takes four integer parameters — the row and column of the first queen on a chessboard, followed by the row and column of the second queen — and generates and returns an 8-by-8 two-dimensional array of `boolean` values in which the elements that correspond to both queens' positions and all the positions reachable by one or both queens are set to `true`, and all the other elements are set to `false`. In writing this method, do not duplicate your code from `oneQueen`; instead, call that method twice, once for each queen, and combine the two boards returned by these calls. Assume that the method `oneQueen` from Part (a) works as specified, regardless of what you wrote there.

Complete the method `twoQueens` below.

```
/** Returns an 8-by-8 boolean two-dimensional array in which
 *  the element at row1, col1, the element at row2, col2,
 *  and all the elements that represent positions reachable
 *  by a queen placed at row1, col1 as well as positions
 *  reachable by a queen placed at row2, col2 are set to
 *  true, and all the other elements are set to false.
 *  Precondition: 0 <= row1 < 8; 0 <= col1 < 8;
 *                0 <= row2 < 8; 0 <= col2 < 8
 */
public static boolean[][] twoQueens(int row1, int col1,
                                    int row2, int col2)
```

For additional practice, write a `boolean` method that takes an `ArrayList` of positions of several queens and returns `true` if there is a position on the board that none of the queens can reach; otherwise your method should return `false`. Use only two variables of the `boolean[][]` type.