Practice Exam #5

SECTION I

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

1. Consider the following statements.

```
double x = 2.5, y = 1.99;
System.out.println((int)(x/y) + (int)(x*y));
```

What is printed when the statements are executed?

(A) `0`
(B) `3`
(C) `4`
(D) `4.0`
(E) `5`

2. Which of the following expressions will evaluate to `true` when `x` and `y` are `boolean` variables with different values?

I. `(x || y) && (!x || !y)`
II. `(x || y) && !(x && y)`
III. `(x && !y) || (!x && y)`

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

3. Which of the following statements will result in a syntax error?

(A) `String x = "123";`
(B) `Integer x = "123";`
(C) `Object x = "123";`
(D) `String[] x = {"123"};`
(E) None of the above

4. Consider the following code segment.

```
String abc = "AAABBBCCC";
String abc1 = abc.substring(0, abc.length() - 1);

abc1 = abc1.substring(1, abc1.length());
System.out.println(abc.indexOf(abc1));
```

What is printed when the code segment is executed?

(A) `-1`
(B) `0`
(C) `1`
(D) `7`
(E) `StringIndexOutOfBoundsException`

5. Consider the following code segment.

```
int x = < integer value >, y = < another integer value >;

while (x > y && x % y != 0)
{
  x -= y;
}
```

Which of the following conditions will always be `true` after the `while` loop, regardless of the initial values of `x` and `y`?

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

6. In OOP, programmers often arrange classes into inheritance hierarchies instead of implementing isolated classes. Which of the following is NOT a valid reason for doing so?

(A) More general classes at the top of the hierarchy can be extended in the project or reused in other projects.
(B) Methods of a superclass can often be reused in its subclasses without duplication of code.
(C) Objects of different subclasses can be passed as parameters to a method designed to accept objects of the superclass.
(D) Objects of different subclasses can be stored in the same array.
(E) All of the above are valid reasons for using inheritance hierarchies.

7. The following method is intended to remove from `ArrayList<Integer> list` all elements whose value is less than zero.

```
public void removeNegatives(ArrayList<Integer> list)
{
  int i = 0, n = list.size();

  while (i < n)
  {
    if (list.get(i) < 0)
    {
      list.remove(i);
      n--;
    }
    i++;
  }
}
```

For which lists of `Integer` values does this method work as intended?

(A) Only an empty list
(B) Only lists that do not contain negative values in consecutive positions
(C) Only lists in which all the negative values occur before all the positive values
(D) Only lists in which all the positive values occur before all the negative values
(E) All lists

8. Consider the following code segment.

```
int[] arr = {1, 2, 3, 4, 5, 6, 7, 8};

for (int k = 1; k <= 6; k += 2)
{
  arr[7] = arr[k];
  arr[k] = arr[k+1];
  arr[k+1] = arr[7];
}
```

Which of the following represents the contents of `arr` after the code segment has been executed?

(A) 1 3 2 5 4 7 6 6
(B) 1 3 2 5 4 7 6 8
(C) 2 1 4 3 6 5 8 7
(D) 2 1 4 3 6 5 7 8
(E) 2 1 4 3 6 5 7 5

Questions 9 and 10 refer to the method `smile` below.

```
public static void smile(int n)
{
  if (n == 0)
    return;

  for (int k = 1; k <= n; k++)
  {
    System.out.print("smile!");
  }

  smile(n-1);
}
```

9. What is printed when `smile(4)` is called?

(A) `smile!`
(B) `smile!smile!`
(C) `smile!smile!smile!`
(D) `smile!smile!smile!smile!`
(E) `smile!smile!smile!smile!smile!smile!smile!smile!smile!smile!`

10. When `smile(4)` is called, how many times will `smile` actually be called, including the initial call?

(A) 2
(B) 3
(C) 4
(D) 5
(E) 10

11. Consider the following code segment.

```
Integer year = 2020;
System.out.println(year.compareTo(2021));
```

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

(A) Syntax error
(B) A positive integer is printed
(C) A negative integer is printed
(D) `true` is printed
(E) `false` is printed

12. Consider the following method.

```
private int swap(int a, int b)
{
  if (a < b)
  {
    b = a;
    a = b;
  }
  return b - a;
}
```

What are the values of the variables `a`, `b`, and `c` after the following statements are executed?

```
int a = 2, b = 5;
int c = swap(a, b);
```

(A) 2, 5, 0
(B) 2, 5, 3
(C) 2, 5, -3
(D) 2, 2, 0
(E) 5, 2, 3

13. What is printed when the following code segment is executed?

```
int sum = 0, d = -1;

for (int count = 10; count > 0; count--)
{
  sum += d;

  if (d > 0)
  {
    d++;
  }
  else
  {
    d--;
  }
  d = -d;
}

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

(A) 0
(B) 5
(C) -5
(D) 10
(E) -10

14. Consider the following incomplete method, which shuffles a list of `Card` objects so that any card can end up at any index in the list `deck` with equal probability.

```
public void shuffle(ArrayList<Card> deck)
{
  int n = deck.size();
  while (n > 1)
  {
    < missing statement >

    Card temp = deck.set(k, deck.get(n-1));
    deck.set(n-1, temp);
    n--;
  }
}
```

Which of the following can replace < *missing statement* > so that the method works as intended?

(A) `int k = Math.random(n);`
(B) `int k = Math.random(deck.size());`
(C) `int k = Math.random(deck.size() - 1);`
(D) `int k = (int)(n * Math.random());`
(E) `int k = (int)((n-1) * Math.random());`

15. The method

```
public void addStars(ArrayList<String> words)
```

is intended to append `"*"` at the end of each string in `words`. Which of the following implementations of `addStars` will work?

I.
```
for (String word : words)
  word += "*";
```

II.
```
for (int k = 0; k < words.size(); k++)
  words.set(k, words.get(k) + "*");
```

III.
```
for (int k = 0; k < words.size(); k++)
  words.add(words.remove(k) + "*");
```

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

16. Suppose a class `Solid` has a method `getVolume()` that returns 0. Two classes, `Cube` and `Pyramid`, extend `Solid`. Which Java feature makes it possible for the following code segment to print the correct values for the volume of a pyramid and a cube?

```
Solid[] solids = new Solid[2];
Solids[0] = new Cube(100);
Solids[1] = new Pyramid(150, 100);

System.out.println("Cube: " + solids[0].getVolume());
System.out.println("Pyramid: " + solids[1].getVolume());
```

(A) abstraction
(B) encapsulation
(C) polymorphism
(D) modularity
(E) method overloading

17. Consider the following method.

```
public boolean examine(String[] letters)
{
  int count = 0;

  for (String letter1 : letters)
  {
    for (String letter2 : letters)
    {
      if (letter1.equals(letter2))
        count++;
    }
  }

  return count > 0;
}
```

What will `examine` return for the following arrays?

```
String[] letters1 = {"A", "B", "C"};
String[] letters2 = {"A", "B", "B"};
String[] letters3 = {"A", "A", "B"};
```

	letters1	letters2	letters3
(A)	true	true	true
(B)	true	true	false
(C)	false	true	true
(D)	false	false	true
(E)	false	false	false

18. Consider the following classes.

```
public class APTestResult
{
  private String subject;
  private int score;

  public int getScore()
  {
    return score;
  }

  /* constructors and other methods not shown */
}

public class APScholar
{
  private String name;
  private int id;
  private ArrayList<APTestResult> exams;

  public int getExamResult(k)
  {
    return exams.get(k);
  }

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

Given

```
APScholar[] list = new APScholar[100];
```

which of the following expressions correctly represents the third AP score of the first `APScholar` in `list`?

(A) `list[0].exams[2].getScore()`
(B) `list[2].exams.getScore()`
(C) `list[2].getExamResult().getScore()`
(D) `list[0].getExamResult().getScore(2)`
(E) None of the above

19. What is printed when the following code segment is executed?

```
String[] xy = {"X", "Y"};
String[] yx = xy;
yx[0] = xy[1];
yx[1] = xy[0];

System.out.println(xy[0] + xy[1] + yx[0] + yx[1]);
```

(A) XXXX
(B) YYYY
(C) XYYX
(D) XYXY
(E) XYYY

20. Consider the following class `Toy`.

```
public class Toy
{
  private String name;

  public Toy (String nm) { name = nm; }
  public String toString() { return name; }

  public void meet(Toy other)
  {
    System.out.println("Hello " + other);
  }
}
```

Suppose the class `Barbie` has a no-arguments constructor. Which of the following code segments compile without syntax errors if `Barbie` extends `Toy`, but cause a syntax error if `Barbie` does not extend `Toy`?

I.
```
Toy kenny = new Toy("Ken");
Barbie barb = new Barbie();
kenny.meet(barb);
```

II.
```
Toy barb = new Barbie();
Toy kenny = new Toy("Ken");
barb.meet(kenny);
```

III.
```
Barbie barb = new Barbie();
Barbie barb2 = new Barbie();
barb.meet(barb2);
```

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

21. Suppose `m` is a two-dimensional array of size 4 by 4, with all its elements initialized to zeros. The method `fill` is defined as follows:

```
public void fill(int[][] m)
{
  int n = m.length;

  for (int i = 1; i < n - 1; i++)
  {
    for (int j = 1; j < n - 1; j++)
      m[i][j] = 1;
  }
}
```

What values will be stored in `m` after `fill(m)` is called?

(A)
```
0000
0000
0000
0000
```

(B)
```
1100
1100
0000
0000
```

(C)
```
0000
0110
0110
0000
```

(D)
```
1110
1110
1110
0000
```

(E)
```
1111
1111
1111
1111
```

22. Classes `Salsa` and `Swing` are subclasses of `Dance`. If the calls

```
perform(new Salsa());
perform(new Swing());
```

are both valid, which of the following headers of the `perform` method(s) in the `DanceSchool` class will compile successfully?

I. Two methods:

```
public void perform(Salsa dance)
public void perform(Swing dance)
```

II.
```
public void perform(Dance dance)
```

III.
```
public void perform(Object dance)
```

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

23. Suppose a, b, and c are positive integers under 1000, and x satisfies the formula

$$\frac{a}{b} = \frac{c}{x}$$

The integer value d is obtained by truncating x to an integer. Which of the following code segments correctly calculate d?

I.
```
d = c * b / a;
```

II.
```
int temp = c * b;
d = temp / a;
```

III.
```
int temp = b / a;
d = c * temp;
```

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

24. Suppose a class `Particle` has the following fields defined.

```
public class Particle
{
  public static final int START_POS = 100;
  private double velocity;

  private boolean canMove()
  { /* implementation not shown */ }

  /* constructors, other fields and methods not shown */
}
```

Which of the following is true?

(A) Java syntax rules wouldn't allow us to use the name `startPos` instead of `START_POS`.

(B) Java syntax rules wouldn't allow us to make `velocity` public.

(C) Both `velocity` and `START_POS` can be changed by one of `Particle`'s methods.

(D) A statement

```
double pos = START_POS + velocity;
```

in `Particle`'s `canMove` method would result in a syntax error.

(E) None of the above

25. Consider the following method.

```
private double compute(int x, int y)
{
  double r = 0;

  if (!(y == 0 || x/y <= 2))
    r = 1 / ((x - 2*y) * (2*x - y));

  return r;
}
```

For which of the following values of `x` and `y` will `compute(x, y)` throw an exception?

(A) `x = 0, y = 0`
(B) `x = 1, y = 2`
(C) `x = 2, y = 1`
(D) `x = 3, y = 5`
(E) None of the above

26. Consider the following method.

```
/** Returns the number of zeros in s.
 *  Precondition: s.length() = 31; s consists of several
 *                0's followed by several 1's
 *                (s can also be all 0's or all 1's).
 */
public int countZeros(String s)
{
  int i = 0, j = 30;
  while (i <= j)
  {
    int k = (i + j) / 2;
    if (s.substring(k, k+1).equals("0"))
      i = k + 1;
    else
      j = k - 1;
  }
  return i;
}
```

How many iterations through the `while` loop will be made in the best and the worst case?

	Best case	Worst case
(A)	1	5
(B)	4	5
(C)	5	5
(D)	1	15
(E)	4	15

27. Consider the following code segment.

```
int x = (int)Math.pow(2, 5);

for (int k = 1; k <= 3; k++)
  x *= (2*x);

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

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

(A) Syntax error: "method pow in class Math cannot be applied to given types"
(B) `3125000` is printed ($= 5^8 \cdot 2^3$)
(C) `8388608` is printed ($= 2^{23}$)
(D) `140737488355328` is printed ($= 2^{47}$)
(E) None of the above

28. Consider the following method.

```
public String encrypt(String word)
{
  int pos = word.length() / 2;

  if (pos >= 1)
  {
    word = encrypt(word.substring(pos)) +
           encrypt(word.substring(0, pos));
  }

  return word;
}
```

Which of the following strings is returned by `encrypt("SECRET")`?

(A) `TERCES`
(B) `TSECRE`
(C) `RETSEC`
(D) `CESTER`
(E) `ETRECS`

29. Consider the following code segment.

```
int[] nums = new int[8];
nums[0] = 0;
int n = 1;

while (n < nums.length)
{
  int k;

  for (k = n; k < 2*n; k++)
    nums[k] = nums[k-n] + 1;

  n = k;
}
```

Which of the following represents the contents of `nums` after the code segment has been executed?

(A) `0 1 1 1 1 1 1 1`
(B) `0 1 0 1 0 1 0 1`
(C) `0 1 1 2 1 2 2 3`
(D) `0 1 2 3 1 2 3 4`
(E) `0 1 2 3 4 5 6 7`

30. Consider the following class.

```
public class Game
{
  private static int bestScore;
  private int score;
  private String player;

  < constructors and methods not shown >
}
```

Which of the following constructors or methods in `Game` will cause a syntax error?

I.
```
public static void resetScore()
{ score = 0; bestScore = 0; }
```

II.
```
public Game()
{ score = 0; bestScore = 0; }
```

III.
```
public void setPlayer(String name)
{ player = name; }
```

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

31. What is printed when the following code segment is executed?

```
int[] factors = {2, 3, 5};
ArrayList<Integer> products = new ArrayList<Integer>();
products.add(1);

for (int f : factors)
{
  int n = products.size();
  for (int k = 0; k < n; k++)
    products.add(f * products.get(k));
}

int n = products.size();
System.out.print(n + " " + products.get(n-2) +
                                  " " + products.get(n-1));
```

(A) 4 3 5
(B) 4 6 15
(C) 8 10 15
(D) 8 10 30
(E) 8 15 30

Questions 32 and 33 refer to the following class `SortX`.

```
public class SortX
{
  public static void sort(String[] items)
  {
    int n = items.length;

    while (n > 1)
    {
      sortHelper(items, n - 1);
      n--;
    }
  }

  private static void sortHelper(String[] items, int last)
  {
    int m = last;

    for (int k = 0; k < last; k++)
    {
      if (items[k].compareTo(items[m]) > 0)
        m = k;
    }

    String temp = items[m];
    items[m] = items[last];
    items[last] = temp;
  }
}
```

32. Suppose `names` is an array of `String` objects:

```
String[] names =
          {"Dan", "Ada", "Claire", "Evan", "Boris"};
```

If `SortX.sort(names)` is running, what is the order of the values in `names` after two complete iterations through the `while` loop in the `sort` method?

(A) `"Boris", "Ada", "Claire", "Dan", "Evan"`
(B) `"Ada", "Claire", "Boris", "Dan", "Evan"`
(C) `"Ada", "Boris", "Claire", "Evan", "Dan"`
(D) `"Ada", "Claire", "Dan", "Evan", "Boris"`
(E) None of the above

33. If `items` contains five values and `SortX.sort(items)` is called, how many times, total, will `items[k].compareTo(items[m])` be called in the `sortHelper` method?

(A) 5
(B) 10
(C) 15
(D) 25
(E) Depends on the values in `items`

34. Consider the following method.

```
/** Returns the number of times the digit d occurs in the
 *  decimal representation of n.
 *  Precondition: n and d are non-negative integers.
 */
private int findDigit(int n, int d)
{
  int count = 0;

  < statement 1 >

  while (n > 0)
  {
    if (n % 10 == d)
    {
      count++;
    }
    < statement 2 >
  }

  return count;
}
```

Which of the following could replace < *statement 1* > and < *statement 2* > to make `findDigit` work as described in the comment for this method?

	< *statement 1* >	< *statement 2* >
(A)	`if (n == 0) return 1;`	`n /= 10;`
(B)	`if (n == 0) return 1;`	`d *= 10;`
(C)	`if (d == 0) count++;`	`n -= n % 10;`
(D)	`if (n == 0 && d == 0) count++;`	`n /= 10;`
(E)	`if (n == 0 && d != 0) return 0;`	`n *= 10;`

Questions 35-37 involve reasoning about classes and objects used in an implementation of a library catalog system.

An object of the class `BookInfo` represents information about a particular book, and an object of the class `LibraryBook` represents copies of a book on the library's shelves.

```
public class BookInfo
{
  private String title;
  private String author;
  private int numPages;

  < constructors not shown >

  public String toString()
  {
    return title + " by " + author;
  }

  public String getTitle() { return title; }

  public int getNumPages() { return numPages; }
}

public class LibraryBook
{
  private BookInfo info;
  private int numCopies;   // Number of copies on shelf

  < constructors not shown >

  public int getNumCopies() { return numCopies; }

  public void setNumCopies(int num) { numCopies = num; }

  public BookInfo getInfo() { return info; }

  /** If there are copies on shelf, decrements
   *  the number of copies left and returns true;
   *  otherwise returns false.
   */
  public boolean checkOut() { /* implementation not shown */ }
}
```

35. If `catalog` is declared in a client class as

```
LibraryBook[] catalog;
```

then which of the following statements will correctly print

title by *author*

for the third book in `catalog`?

I. `System.out.println(catalog[2]);`

II. `System.out.println(catalog[2].getInfo());`

III. `System.out.println(catalog[2].getInfo().toString());`

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

36. Consider the following method from another class, a client of `LibraryBook`.

```
/** Returns the total number of pages in all
 *  books in catalog that are on the shelves.
 */
public int totalPages(LibraryBook[] catalog)
{
  int count = 0;

  for (LibraryBook bk : catalog)
  {
    < statement >
  }
  return count;
}
```

Which of the following replacements for < *statement* > completes the method as intended?

(A) `count += bk.numCopies * bk.info.numPages;`
(B) `count += bk.getNumCopies() * bk.getNumPages();`
(C) `count += bk.(numCopies * info.getNumPages());`
(D) `count += bk.getNumCopies() * bk.getInfo().getNumPages();`
(E) None of the above

37. Which of the following code segments will correctly complete the `checkOut()` method of the `LibraryBook` class?

I.
```
if (getNumCopies() == 0)
{
  return false;
}
else
{
  setNumCopies(getNumCopies() - 1);
  return true;
}
```

II.
```
int n = getNumCopies();
if (n == 0)
{
  return false;
}
else
{
  setNumCopies(n - 1);
  return true;
}
```

III.
```
if (numCopies == 0)
{
  return false;
}
else
{
  numCopies--;
  return true;
}
```

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

38. Consider the following method.

```
public int[][] makeCounts(int n)
{
  int[][] counts = new int[3][n];
  counts[0][0] = 0;
  counts[1][0] = 0;
  counts[2][0] = 1;

  for (int k = 1; k < n; k++)
  {
    counts[0][k] = counts[0][k-1] + counts[1][k-1];
    counts[1][k] = counts[1][k-1] + counts[0][k-1] +
                                    counts[2][k-1];
    counts[2][k] = counts[2][k-1] + counts[1][k-1];
  }
  return counts;
}
```

What values are in the array returned by `makeCounts(5)`?

(A)

```
0  0  1  1  1
0  1  1  1  1
1  1  1  1  1
```

(B)

```
0  0  1  2  3
0  1  2  3  4
1  2  3  4  5
```

(C)

```
0  0  1  3  6
0  1  2  5  8
1  1  4  7  9
```

(D)

```
0  0  1  3  8
0  1  2  5 12
1  1  2  4  9
```

(E)

```
0  1  3  9 27
0  1  3  9 27
1  1  3  9 27
```

39. What is printed when the following code segment is executed?

```
ArrayList<Integer> lst = new ArrayList<Integer>();

for (int k = 1; k <= 6; k++)
  lst.add(k);

for (int k = 0; k < 3; k++)
{
   Integer i = lst.remove(k);
   lst.add(i);
}
for (Integer i : lst)
  System.out.print(i);
```

(A) 123456
(B) 246135
(C) 456123
(D) 456321
(E) IndexOutOfBoundsException

40. Consider the following method.

```
public int mysteryCount(int[] p)
{
  int count = 0;
  for (int i = 0; i < p.length; i++)
  {
    count++;
    int j = p[i];
    while (j != i)
    {
      j = p[j];
      count++;
    }
  }
  return count;
}
```

Given

```
int[] arr = {0, 2, 3, 1};
```

what will `mysteryCount(arr)` return?

(A) 2
(B) 3
(C) 5
(D) 10
(E) 13

Practice Exam #5

SECTION II

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

1. The class `Cipher` provides methods to encrypt and decrypt a message using a simple substitution cipher.

```
public class Cipher
{
  private static final String abc = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

  /** Returns a new string as described in part (a).
   *  Precondition: All letters in text are uppercase;
   *                0 < key > 26.
   */
  public static String encrypt(String text, int key)
  { /* to be implemented in part (a) */ }

  /** Returns a new string as described in part (b) */
  public static String decrypt(String code, int key)
  { /* to be implemented in part (b) */ }
}
```

(a) Write a method `encrypt` that takes a string `text` and a positive integer `key` as parameters and replaces each letter in `text` with the letter in the `abc` string shifted by `key` positions. If the index of the substitution letter exceeds 25, then wrap-around occurs back through the beginning of the `abc` string. For example, if `key` is 3, then 'A' becomes 'D', 'B' becomes 'E', ..., 'W' becomes 'Z', 'X' becomes 'A', 'Y' becomes 'B', and 'Z' becomes 'C'. Assume that all letters in `text` are uppercase. All the characters that are not letters remain unchanged.

Complete the method `encrypt` below.

```
/** Returns a new string in which each letter in text is
 *  replaced with the letter whose index in abc is equal
 *  to the index of the original letter plus key, with
 *  wraparound, if needed, back to the beginning of abc.
 *  All the characters in text that are not letters remain
 *  unchanged.
 *  Precondition: all letters in text are uppercase;
 *                0 < key < 26.
 */
public static String encrypt(String text, int key)
```

(b) Write a method `decrypt` that restores the original message from the encrypted message generated by the `encrypt` method above. For example, if `key` is 3, then 'B' in the encrypted message corresponds to 'Y' in the original message. Notice that the decryption algorithm is exactly the same as the encryption algorithm if you use a modified key, so there is no need to replicate the code from the `encrypt` method from Part (a). Assume that the `encrypt` method in Part (a) works as specified, regardless of what you wrote there.

Complete the method `decrypt` below.

```
/** Returns a new string in which each letter in code is
 *  replaced with the letter whose index in abc is equal
 *  to the index of the letter minus key, with
 *  wrap-around at index 0.
 *  Precondition: All letters in code are uppercase;
 *                0 < key < 26
 */
public static String decrypt(String code, int key)
```

For additional practice, modify the `Cipher` class to accommodate both uppercase and lowercase letters in `text` and `code`, preserving the case of each letter. For even more practice, find on the Internet the description of the Vigenère cipher and implement the `encrypt` and `decrypt` methods for it.

2. The College Board gives awards to students who are successful on AP exams:

   - **AP Scholar**: Received grades of 3 or higher on three or more AP Exams
   - **AP Scholar with Honor**: Received an average grade of at least 3.25 on all AP Exams taken, and grades of 3 or higher on four or more of these exams
   - **AP Scholar with Distinction**: Received an average grade of at least 3.5 on all AP Exams taken, and grades of 3 or higher on five or more of these exams

   These awards are summarized in the following table:

	AP Scholar	AP Scholar with Honor	AP Scholar with Distinction
Average grade on all AP exams taken, at least	No effect	3.25	3.5
Minimum grade that counts toward an award	3	3	3
Minimum number of exams not below the minimum grade	3	4	5

   Notice that a student may receive low grades on some AP exams and still qualify for any one of the above awards.

   The class `APStudent` provides methods that help to determine the award level for a student:

   - `addExam` — takes a given grade (an `int`) as a parameter and updates this `APStudent`'s instance variables
   - `awardLevel` — returns the award level of this `APStudent` based on all the added exams

   The award level returned by `awardLevel` is represented by an integer: 0 for no award, 1 for AP Scholar, 2 for AP Scholar with Honor, and 3 for AP Scholar with Distinction.

   The `APStudent` class does not define a constructor — it relies on the default no-arguments constructor.

For example, the statements

```
APStudent s = new APStudent();
s.addExam(2);
System.out.print(s.awardLevel() + " ");
s.addExam(4);
s.addExam(5);
System.out.print(s.awardLevel() + " ");
s.addExam(3);
System.out.print(s.awardLevel() + " ");
s.addExam(4);
System.out.print(s.awardLevel() + " ");
s.addExam(3);
System.out.print(s.awardLevel() + " ");
s.addExam(1);
System.out.print(s.awardLevel() + " ");
System.out.println();
```

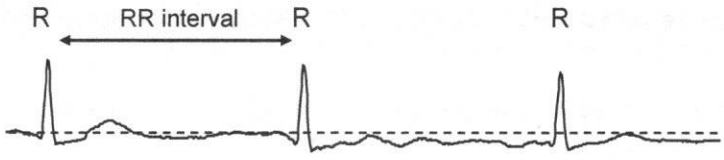
print

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

Write the complete `APStudent` class, including the required instance variables and the two methods described above. Your implementation must meet the specifications and conform to the above example. **Do not use arrays, `ArrayLists`, or other data structures in your solution.**

For additional practice, write a class `APStudentWithName` as a subclass of `APStudent`. Supply a constructor that takes the student's name as a parameter and a `toString` method that returns that name. Write and test a method of a client class of `APStudentWithName` that takes an `ArrayList` of `APStudentWithName` objects and returns an `ArrayList` of names of the students from the list who qualify for the AP Scholar with Distinction award.

3. In computerized monitoring of an electrocardiogram (also called ECG or EKG) one of the tasks is detecting the most prominent spikes, called R-peaks, in the signal.



Measuring the time intervals between consecutive R-peaks allows the ECG device to compute the patient's heart rate.

In this question you will write two methods of the class `ECG` that help make a list of all R-peaks and calculate the patient's heart rate.

In a computer system, the ECG voltage signal is digitized at a certain sampling rate and the resulting values are stored in a "voltage" array. For example, a sampling rate of 300 means that 300 samples are obtained and stored every second. An integer constant `SAMPLING_RATE` in the `ECG` class indicates how many consecutive elements in the voltage array represent one second of an ECG recording.

We are not concerned with the magnitudes of R-peaks here, only with the times when they occur. Knowing the sampling rate, we can compute the time interval between two R-peaks from their indices in the voltage array.

A partial definition of the `ECG` class is shown below.

```
public class ECG
{
  /** ECG sampling rate in samples taken per second. */
  private static final int SAMPLING_RATE = 300;

  /** Minimum possible distance between R-peaks. */
  private static final int DELTA = SAMPLING_RATE / 10;

  /** Returns true if an R-peak is detected at index k;
   *  otherwise returns false.
   */
  private static boolean isRpeak(double[] v, int k)
  { /* implementation not shown */ }

  /** Returns an ArrayList of indices of all consecutive
   *  R-peaks in v, as described in part (a).
   */
  public static ArrayList<Integer> findRpeaks(double[] v)
  { /* to be implemented in part (a) */ }
```

Continued

```
    /** Returns the heart rate in beats per minute obtained from
     *  a list of indices of R-peaks.
     *  Precondition: rPeakPositions.size() >= 2
     */
    public static int heartRate(ArrayList<Integer> rPeakPositions)
    { /* to be implemented in part (b) */ }

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

(a) Write the method `findRpeaks(double[] v)`. This method returns an `ArrayList` of the indices of all R-peaks found in a given voltage array `v`, preserving their order in time. Call `isRpeak(v, k)` to decide whether there is an R-peak in `v` at index `k`. The implementation of the algorithm used by `isRpeak` is not shown. This algorithm examines the values in `v` in a certain interval around `k`. Start looking for R-peaks at `k = DELTA` and finish at `k = v.length-1-DELTA` to leave room for `isRpeak` to do its job. Also, when an R-peak is detected at index `k`, look for the next R-peak starting from `k+DELTA` (to avoid detecting the same R-peak multiple times).

Complete the method `findRpeaks` below.

```
/** Returns an ArrayList of indices of all consecutive
 *  R-peaks in v between v[DELTA] and v[v.length-1-DELTA].
 *  An R-peak is determined by a call to the method isRpeak.
 *  R-peaks must be at least DELTA apart.
 */
public static ArrayList<Integer> findRpeaks(double[] v)
```

(b) Write the method `heartRate` that takes a list of indices of R-peaks in the voltage array and returns the patient's heart rate in beats per minute, rounded to the nearest integer. The heart rate is derived from the average length of RR-intervals (distances between consecutive R-peaks) in the list. To obtain the average distance, take the distance between the first and the last R-peak (the difference between the last and the first elements of the list) and divide it by the number of RR-intervals represented by the list.

Use `SAMPLING_RATE` to convert the indices into real time. For example, if the average length of RR-intervals is 250 (indices) and `SAMPLING_RATE` is 300 (indices/second), then the average length of the RR-interval is $250/300 = 5/6$ seconds, which corresponds to $60 \cdot 6 / 5 = 72$ beats per minute.

Complete the method `heartRate` below.

```
/** Returns the heart rate in beats per minute obtained from
 *  rPeakPositions, the list of the indices of R-peaks.
 *  Precondition: rPeakPositions.size() >= 2
 */
public static int heartRate(ArrayList <Integer> rPeakPositions)
```

For additional practice, implement the `isRpeak(int k)` method. This method returns `true` if there is an R-peak at index `k` in the voltage array `v`; otherwise it returns `false`. Use the following simplified algorithm for detecting R-peaks: there is an R-peak at `k` in `v` if the following three conditions are satisfied:

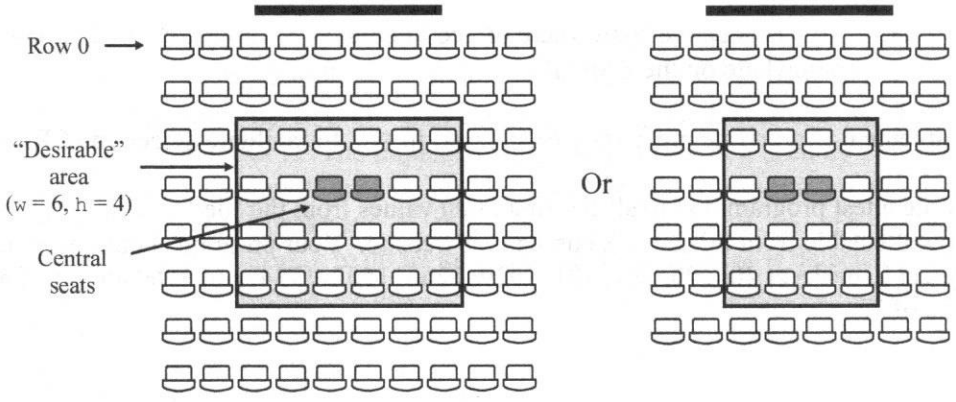
1. `v[k] > 1.0`
2. `v[k]` is equal to the maximum value among `v[j]` for all `j` in the interval `k - DELTA ≤ j ≤ k + DELTA`, where `DELTA` is a constant defined in `ECG`.
3. `v[j]` is negative somewhere on the interval `k - DELTA ≤ j ≤ k` and also somewhere on the interval `k ≤ j ≤ k + DELTA`.

Implement two private methods `max` and `min` in `ECG` to facilitate detection of R-peaks.

Write a test program that reads the first 2000 values from the file `"ecg.dat"`, downloadable from this book's companion website. Your program should be able to detect R-peaks at 40, 290, 541, 791, 1029, 1266, 1502, 1734, and 1969 and yield a heart rate of 75.

4. Seats in a movie theater are modeled by a two-dimensional array of `boolean` values: `true` represents an occupied seat and `false` represents a vacant seat. Row number 0 represents the row of seats closest to the screen.

In Part (a) we assume that the most desirable seats are in the rectangular area of width `w` and "height" `h`, centered in front of the screen (middle of the row), in the rows numbered `2` to `h+1`. Both `w` and the number of seats in each row are even numbers. For example:



You will write a method that returns the number of vacant seats in the "desirable" area.

In Part (b), we introduce a different idea: the "best two seats" are in the center of the central row of the seats array. If the number of rows in the theater is even, and there are two "central" rows, then the best seats are in the one that is closer to the screen. (In the above examples, the two central seats happen to be within the "desirable" area, but this is not required.) You will write a method that finds a pair of adjacent vacant seats that are closest to the "best two seats."

(a) Write a static method `vacantGoodSeats` that takes a two-dimensional array representing the seats in a movie theater and `w` and `h`, the width and "height" of the "desirable" seating area, and returns the number of vacant seats in that area.

Complete the method `vacantGoodSeats` below.

```
/** Returns the number of vacant seats in the "desirable"
 *  area of the seats array: the h rows starting from row 2
 *  and the w contiguous seats in the center of each row.
 *  Precondition: The entire "desirable" area fits within
 *                the seats array; w and the number of seats
 *                in each row are even numbers.
 */
public static int vacantGoodSeats(boolean[][] seats,
                                  int w, int h)
```

(b) The class `Location` has a constructor that takes two parameters, `row` and `col`, and a method

```
public double distanceFrom(Location other)
```

that returns the distance from this location to `other`. Write a method `bestTwoSeats` that takes a two-dimensional array of seats, as described above, and finds the location of a pair of adjacent vacant seats closest to the center of the theater. The distance is measured from the left seat in the found pair of seats to the left seat in the central pair of seats. `bestTwoSeats` returns the location of the left seat in the found pair. If a pair of adjacent vacant seats is not found, `bestTwoSeats` returns `null`.

Complete the method `bestTwoSeats` below.

```
/** Returns the location of a pair of vacant seats closest
 *  to the central pair of seats. The central pair is
 *  in the middle of the center row; if the number of rows
 *  is even, the "center" row is the one closer to the
 *  screen. The location of a pair of seats is defined
 *  as the row and column of the left seat in the pair
 *  (the seat with the smaller column index). Returns
 *  null if a pair of adjacent vacant seats is not found.
 */
public static Location bestTwoSeats(boolean[][] seats)
```

For additional practice, implement the `Location` class. Make the distance between seats equal to the absolute value of the difference in rows plus the absolute value of the difference in columns.