AP Computer Science
1/21/2016

Mr. Brett Wortzman
Issaquah High School

Name: ______________________________

# Semester 1 Final Exam

GradeCam ID

| | | | |
|---|---|---|---|
| | | | |
| ⓪ | ⓪ | ⓪ | ⓪ |
| ① | ① | ① | ① |
| ② | ② | ② | ② |
| ③ | ③ | ③ | ③ |
| ④ | ④ | ④ | ④ |
| ⑤ | ⑤ | ⑤ | ⑤ |
| ⑥ | ⑥ | ⑥ | ⑥ |
| ⑦ | ⑦ | ⑦ | ⑦ |
| ⑧ | ⑧ | ⑧ | ⑧ |
| ⑨ | ⑨ | ⑨ | ⑨ |

| | | | | | |
|---|---|---|---|---|---|
| 1. | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |
| 2. | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |
| 3. | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |
| 4. | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |
| 5. | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |
| 6. | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |
| 7. | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |
| 8. | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |
| 9. | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |
| 10. | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |
| 11. | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |
| 12. | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |
| 13. | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |
| 14. | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |
| 15. | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |
| 16. | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |
| 17. | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |
| 18. | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |
| 19. | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |
| 20. | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |

**Section I Total: ____ / 40**

Section II – Free Response

1) (a) _____ / 5

   (b) _____ / 4

   *Total: _____ / 9*

2) (a) _____ / 4

   (b) _____ / 5

   *Total: _____ / 9*

**Section II Total: _____ / 18 * 2.222 = _____ / 40**

## Overall Total:

**_____ / 80**

## AP Exam Score:

**_______**

***(Turn Over)***

## Section I – Multiple Choice (40 points total)
Suggested Time: 45 minutes

1) What is output by the following line of code?

```
System.out.println("\"A jar\n of almonds\"");
```

(A) `\A jar\n of almonds\`
(B) `"A jar of almonds"`
(C)
```
A jar
of almonds
```
(D)
```
\"A jar
of almonds\"
```
(E)
```
"A jar
 of almonds"
```

2) Assume a method exists with the following prototype:

```
public static int getValue(int n)
```

Consider the following possible overloads of the method **getValue**:

```
  I.  public static int getValue(double n)
 II.  public static double getValue(int n)
III.  public static int getValue(int n, int x)
```

Which of the above overloads is invalid?

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

3) Which of the following statements about constructors is NOT true?

(A) All constructors must have the same name as the class they are in.
(B) A class may have a constructor that takes no parameters.
(C) Constructors' return type must be declared `void`.
(D) A constructor is invoked when a program creates an object with the `new` operator.
(E) Constructors may be overloaded in the same way as other methods.

4) Consider the following output:

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

Which of the following code segments will produce this output?

(A)
```
for (int j = 1; j <= 5; j++) {
   for (int k = j; k <= 5; k++) {
      System.out.print(k + " ");
   }
   System.out.println();
}
```

(B)
```
for (int j = 1; j <= 5; j++) {
   for (int k = 1; k <= j; k++) {
      System.out.print(j + " ");
   }
   System.out.println();
}
```

(C)
```
for (int j = 1; j <= 5; j++) {
   for (int k = 5; k >= j; k--) {
      System.out.print(j + " ");
   }
   System.out.println();
}
```

(D)
```
for (int j = 1; j <= 5; j++) {
   for (int k = 5; k >= 1; k--) {
      System.out.print(j + " ");
   }
   System.out.println();
}
```

(E)
```
for (int j = 1; j <= 5; j++) {
   for (int k = 1; k <= 5; k++) {
      System.out.print(j + " ");
   }
   System.out.println();
}
```

5) Consider the following segment of code:

```
String word = “conflagaration”;
int x = word.indexOf(“flag”);
String s = word.substring(0, x);
```

What will be the result of executing the above segment?

(A) String `s` will contain `"con"`.
(B) String `s` will contain `"conf"`.
(C) String `s` will contain `"flag"`.
(D) String `s` will be the empty string.
(E) A syntax error will occur.

6) Consider the following class:

```
public class Customer {
    private String myName;
    private String myPhone;
    private int myId;

    // constructor and accessors getName, getPhone, getID not shown
}
```

Assume that a client method has an array `customers` that has been initialized with `Customer` objects. Which of the following will correctly print out the *name only* of each `Customer` in the array?

I.
```
for (int i = 0; i < customers.length; i++) {
    System.out.println(customers[i].myName);
}
```
II.
```
for (int i = 0; i < customers.length; i++) {
    System.out.println(customers[i].getName());
}
```
III.
```
for (Customer c : customers) {
    System.out.println(c.getName());
}
```

(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[] oldArray = {1, 2, 3, 4, 5, 6, 7, 8, 9};
int[][] newArray = new int[3][3];

int row = 0;
int col = 0;
for (int i = 0; i < oldArray.length; i++) {
    int value = oldArray[i];
    newArray[row][col] = value;
    row++;
    if ((row % 3) == 0) {
        col++;
        row = 0;
    }
}

System.out.println(newArray[0][2]);
```

What is printed as a result of executing the code segment?

(A) `3`
(B) `4`
(C) `5`
(D) `7`
(E) `8`

8) Consider the following code segment:

```
if (n == 3)
    k -= 3;
else if (n == 1)
    k--;
```

Suppose that the given segment is rewritten in the form

```
if (/* condition */)
    /* assignment statement */;
```

Given that `n` and `k` are integers and that the rewritten code performs the same task as the original code, which of the following could be used as `/* condition */` and `/* assignment statement */`?

| | | |
|---|---|---|
| (A) | condition: `n == 1 \|\| n == 3` | assignment: `k = n - k` |
| (B) | condition: `n == 1 \|\| n == 3` | assignment: `k -= n` |
| (C) | condition: `n == 1 \|\| n == 3` | assignment: `k -= 3` |
| (D) | condition: `n == 1 && n == 3` | assignment: `k -= n` |
| (E) | condition: `n == 1 && n == 3` | assignment: `k -= 3` |

9) Given the declarations

```
int p = 5, q = 3;
```

which of the following expressions will evaluate to 7.5?

```
I.    (double)p * (double)q / 2;
II.   (double)p * (double)(q / 2);
III.  (double)(p * q / 2);
```

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

10) A square matrix is declared as:

```
int[][] mat = new int[SIZE][SIZE];
```

where `SIZE` is an appropriate integer constant. Consider the following method:

```
public static void mystery(int[][] mat, int value, int top,
                           int left, int bottom, int right) {
    for (int i = left; i <= right; i++) {
        mat[top][i] = value;
        mat[bottom][i] = value;
    }
    for (int i = top + 1; i <= bottom; i++) {
        mat[i][left] = value;
        mat[i][right] = value;
    }
}
```

Assuming that there are no out-of-range errors, which best describes what method `mystery` does?

(A) Places `value` in each element of the border of the rectangle with corners `(top, left)` and `(bottom, right)`.
(B) Places `value` in the topmost and bottommost rows of the rectangle with corners `(top, left)` and `(bottom, right)`.
(C) Places `value` in the corners of the rectangle with corners `(top, left)` and `(bottom, right)`.
(D) Places `value` in the diagonals of the square with corners `(top, left)` and `(bottom, right)`.
(E) Places `value` in each element of the rectangle with corners `(top, left)` and `(bottom, right)`.

11) Consider the following code:

```
public static void changeList(List<Integer> list) {
    int len = list.size()
    for (int i = 0; i < len; i++) {
        list.add(i + 1, i);
        list.set(i, i + 2);
    }
}

public static void main(String[] args) {
    List<Integer> myList = new ArrayList<Integer>();
    list.add(6);
    list.add(1);
    list.add(8);

    changeList(myList);
}
```

What will be in `myList` when `main` is finished executing?

(A) `[6, 1, 8]`
(B) `[8, 1, 6]`
(C) `[6, 0, 1, 2, 1, 8]`
(D) `[2, 3, 4, 2, 1, 8]`
(E) `[2, 3, 4, 2, 1, 6]`

12) Consider the following method:

```
public static int mystery(int[] arr) {
    int x = 0;

    for (int k = 0; k < arr.length; k += 2) {
        x += arr[k];
    }

    return x;
}
```

Assume that the array nums has been declared and initialized as follows:

```
int[] nums = {3, 6, 1, 0, 1, 4, 2};
```

What value will be returned as a result of the call `mystery(nums)`?

(A) `5`
(B) `6`
(C) `7`
(D) `10`
(E) `17`

13) At a certain high school, students receive letter grades based on the following scale:

| Numeric Score | Letter Grade |
|---|---|
| 93 or above | A |
| From 84 to 92 inclusive | B |
| From 75 to 83 inclusive | C |
| Below 75 | F |

Which of the following code segments will assign the correct string to `grade` for a given integer score?

I.
```
if (score >= 93)
   grade = "A";
if (84 <= score <= 92)
   grade = "B";
if (75 <= score <= 83)
   grade = "C";
if (score < 75)
   grade = "F";
```

II.
```
if (score >= 93)
   grade = "A";
else if (score >= 84)
   grade = "B";
else if (score >= 75)
   grade = "C";
else
   grade = "F";
```

III.
```
if (score >= 93)
   grade = "A";
if (score >= 84 && score <= 92)
   grade = "B";
if (score >= 75 && score <= 83)
   grade = "C";
if (score < 75)
   grade = "F";
```

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

Questions 14-15 refer to the **`Worker`** class below:

```
public class Worker {
    private String myName;
    private double myHourlyWage;
    private boolean isUnionMember;

    // constructors and accessors getName, getHourlyWage, getUnionStatus
    //   not shown

    // Permanently increase hourly wage by amt.
    public void incrementWage(double amt)
    { /* implementation of incrementWage */ }

    // Switch value of isUnionMember from true to false and vice versa.
    public void changeUnionStatus()
    { /* implementation of changeUnionStatus */ }
}
```

14) Which of the following is a correct replacement for `/*` *implementation of* `changeUnionStatus */`?

I.
```
isUnionMember = !isUnionMember;
```

II.
```
if (isUnionMember)
   isUnionMember = !isUnionMember;
```

III.
```
if (isUnionMember)
   isUnionMember = false;
else
   isUnionMember = true;
```

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

15) Which of the following is a correct replacement for `/*` *implementation of* `incrementWage */`?

(A) `myHourlyWage = amt;`
(B) `myHourlyWage += amt;`
(C) `return myHourlyWage + amt;`
(D) `getHourlyWage() += amt;`
(E) `return getHourlyWage() + amt;`

16) Consider the following incomplete class declaration.

```
public class TimeRecord {
    private int hours;
    private int minutes; // 0 <= minutes < 60

    /** Adds h hours and m minutes to this TimeRecord.
     *  @param h the number of hours
     *        Assumes h >= 0
     *  @param m the number of minutes
     *        Assumes m >= 0
     */
    public void advance(int h, int m) {
        hours = hours + h;
        minutes = minutes + m;
        /* missing code */
    }

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

Which of the following can be used to replace `/* missing code */` so that `advance` will correctly update the time?

(A) `hours = hours + minutes / 60;`

(B) `minutes = minutes % 60;`

(C) `minutes = minutes + hours % 60;`

(D) `hours = hours + minutes % 60;`
`minutes = minutes / 60;`

(E) `hours = hours + minutes / 60;`
`minutes = minutes % 60;`

17) 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

18) Consider the following method, intended to return `true` if all elements in array `arr` are even numbers and `false` otherwise:

```
public boolean allEven(int[] arr) {
    boolean isEven = /* expression */ ;
    for (int k = 0; k < arr.length; k++) {
        /* loop body */
    }
    return isEven;
}
```

Which of the following replacements for `/* expression */` and `/* loop body */` should be used so that method `allEven` will work as intended?

```
         /* expression */     /* loop body */
    (A)  false                if ((arr[k] % 2) == 0)
                                  isEven = true;
    (B)  false               if ((arr[k] % 2) != 0)
                                  isEven = false;
                              else
                                  isEven = true;
    (C)  true                if ((arr[k] % 2) != 0)
                                  isEven = false;
    (D)  true                if ((arr[k] % 2) != 0)
                                  isEven = false;
                              else
                                  isEven = true;
    (E)  true                if ((arr[k] % 2) == 0)
                                  isEven = false;
                              else
                                  isEven = true;
```

19) An `Insect` class is to be written, containing the following data fields:

- `age`, which will be initialized to 0 when an `Insect` is constructed and incremented each time an `Insect` takes an action
- `direction`, which will be initialized to the direction the `Insect` is facing when placed in the garden.
- `lifespan`, which indicates how many actions an `Insect` can take before it dies and is the same for all `Insect`s
- `position`, which will be initialized to the location in a garden where the `Insect` is placed when it is constructed.

Which field in the `Insect` class should be a class constant?

(A) `age`
(B) `direction`
(C) `lifespan`
(D) `position`
(E) None of these fields should be a class constant

20) Consider the following method, which is intended to return the element of a 2-dimentional array that is closest in value to a specified number, `val`:

```
/** @return the element of 2-dimensional array mat whose value is closest to
 *          val
 */
public double findClosest(double[][] mat, double val) {
    double answer = mat[0][0];
    double minDiff = Math.abs(answer – val);

    for (double[] row : mat) {
        for (double num : row) {
            if ( /* missing code */ ) {
                answer = num;
                minDiff = Math.abs(num – val);
            }
        }
    }
    return answer;
}
```

Which of the following could be used to replace `/* missing code */` so that `findClosest` will work as intended?

(A) `val – row[num] < minDiff`
(B) `Math.abs(num – minDiff) < minDiff`
(C) `val – num < 0.0`
(D) `Math.abs(num – val) < minDiff`
(E) `Math.abs(row[num] – val) < minDiff`

## Section II – Free Response (18 points total – scaled up to 40)
Suggested Time: 45 minutes

*The following question appeared on the 2014 AP Computer Science A Exam.*

1) This question involves reasoning about strings made up of uppercase letters. You will implement two related methods that appear in the same class (not shown). The first method takes a single string parameter and returns a scrambled version of that string. The second method takes a list of strings and modifies the list by scrambling each entry in the list. Any entry that cannot be scrambled is removed from the list.

(a) Write the method `scrambleWord`, which takes a given word and returns a string that contains a scrambled version of the word according to the following rules.

- The scrambling process begins at the first letter of the word and continues from left to right.
- If two consecutive letters consist of an `"A"` followed by a letter that is not an `"A"`, then the two letters are swapped in the resulting string.
- Once the letters in two adjacent positions have been swapped, neither of those two positions can be involved in a future swap.

The following table shows several examples of words and their scrambled versions.

| `word` | Result returned by `scrambleWord(word)` |
|---|---|
| "TAN" | "TNA" |
| "ABRACADABRA" | "BARCADABARA" |
| "WHOA" | "WHOA" |
| "AARDVARK" | "ARADVRAK" |
| "EGGS" | "EGGS" |
| "A" | "A" |
| "" | "" |

Complete method `scrambleWord` below.

```
/** Scrambles a given word.
 *  @param word the word to be scrambled
 *  @return the scrambled word (possibly equal to word)
 *  Precondition: word is either an empty string or contains only uppercase
 *                letters.
 *  Postcondition: the string returned was created from word as follows:
 *   - the word was scrambled, beginning at the first letter and continuing from
 *     left to right
 *   - two consecutive letters consisting of "A" followed by a letter that was
 *     not "A" were swapped
 *   - letters were swapped at most once
 */
public static String scrambleWord(String word)
```

*Continued on next page…*

(b) Write the method **`scrambleOrRemove`**, which replaces each word in the parameter **`wordList`** with its scrambled version and removes any words that are unchanged after scrambling. The relative ordering of the entries in **`wordList`** remains the same as before the call to **`scrambleOrRemove`**.

The following example shows how the contents of **`wordList`** would be modified as a result of calling **`scrambleOrRemove`**.

Before the call to **`scrambleOrRemove`**:

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| **`wordList`** | "TAN" | "ABRACADABRA" | "WHOA" | "APPLE" | "EGGS" |

After the call to **`scrambleOrRemove`**:

| | 0 | 1 | 2 |
|---|---|---|---|
| **`wordList`** | "TNA" | "BARCADABARA" | "PAPLE" |

Assume that **`scrambleWord`** is in the same class as **`scrambleOrRemove`** and works as specified, regardless of what you wrote in part (a). Complete method **`scrambleOrRemove`** below.

```
/** Modifies wordList by replacing each word with its scrambled
 *  version, removing any words that are unchanged as a result of scrambling.
 *  @param wordList the list of words
 *  Precondition: wordList contains only non-null objects
 *  Postcondition:
 *    - all words unchanged by scrambling have been removed from wordList
 *    - each of the remaining words has been replaced by its scrambled version
 *    - the relative ordering of the entries in wordList is the same as it was
 *      before the method was called
 */
public static void scrambleOrRemove(List<String> wordList)
```

*The following question appeared on the 2012 AP Computer Science A Exam.*

2) A grayscale image is represented by a 2-dimensional rectangular array of pixels (picture elements). A pixel is an integer value that represents a shade of gray. In this question, pixel values can be in the range from 0 through 255, inclusive. A black pixel is represented by 0, and a white pixel is represented by 255. The declaration of the `GrayImage` class is shown below. You will write two unrelated methods of the `GrayImage` class.

```
public class GrayImage
{
   public static final int BLACK = 0;
   public static final int WHITE = 255;

   /** The 2-dimensional representation of this image. Guaranteed not to
    *    be null.
    *  All values in the array are within the range [BLACK, WHITE],
    *    inclusive.
    */
   private int[][] pixelValues;

   /** @return the total number of white pixels in this image.
    *  Postcondition: this image has not been changed.
    */
   public int countWhitePixels()
   { /* to be implemented in part (a) */ }

   /** Processes this image in row-major order and decreases the value
    *    of each pixel at position (row, col) by the value of the pixel
    *    at position (row + 2, col + 2) if it exists.
    *  Resulting values that would be less than BLACK are replaced by
    *    BLACK.
    *  Pixels for which there is no pixel at position (row + 2, col + 2)
    *    are unchanged.
    */
   public void processImage()
   { /* to be implemented in part (b) */ }
}
```

*Continued on next page...*

(a) Write the method **countWhitePixels** that returns the number of pixels in the image that contain the value **WHITE**. For example, assume that **pixelValues** contains the following image.

|   | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | **255** | 184 | 178 | 84 | 129 |
| 1 | 84 | **255** | **255** | 130 | 84 |
| 2 | 78 | **255** | 0 | 0 | 78 |
| 3 | 84 | 130 | **255** | 130 | 84 |

A call to `countWhitePixels` method would return 5 because there are 5 entries (shown in boldface) that have the value `WHITE`.

Complete method **countWhitePixels** below.

```
/** @return the total number of white pixels in this image.
 *   Postcondition: this image has not been changed.
 */
public int countWhitePixels()
```

*Continued on next page...*

(b) Write the method **processImage** that modifies the image by changing the values in the instance variable **pixelValues** according to the following description. The pixels in the image are processed one at a time in row-major order. Row-major order processes the first row in the array from left to right and then processes the second row from left to right, continuing until all rows are processed from left to right. The first index of **pixelValues** represents the row number, and the second index represents the column number.

The pixel value at position (row, col) is decreased by the value at position (row + 2, col + 2) if such a position exists. If the result of the subtraction is less than the value **BLACK**, the pixel is assigned the value of **BLACK**. The values of the pixels for which there is no pixel at position (row + 2, col + 2) remain unchanged. You may assume that all the original values in the array are within the range **[BLACK, WHITE]**, inclusive.

The following diagram shows the contents of the instance variable **pixelValues** before and after a call to **processImage**. The values shown in boldface represent the pixels that could be modified in a grayscale image with 4 rows and 5 columns.

Before Call to **processImage**

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | **221** | **184** | **178** | 84 | 135 |
| 1 | **84** | **255** | **255** | 130 | 84 |
| 2 | 78 | 255 | 0 | 0 | 78 |
| 3 | 84 | 130 | 255 | 130 | 84 |

After Call to **processImage**

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | **221** | **184** | **100** | 84 | 135 |
| 1 | **0** | **125** | **171** | 130 | 84 |
| 2 | 78 | 255 | 0 | 0 | 78 |
| 3 | 84 | 130 | 255 | 130 | 84 |

Complete method processImage below.

```
/** Processes this image in row-major order and decreases the value
 *    of each pixel at position (row, col) by the value of the pixel
 *    at position (row + 2, col + 2) if it exists.
 *  Resulting values that would be less than BLACK are replaced by BLACK.
 *  Pixels for which there is no pixel at position (row + 2, col + 2)
 *    are unchanged.
 */
public void processImage()
```

# Appendix A — Java Quick Reference

Accessible Methods from the Java Library That May Be Included on the Exam

**`class java.lang.Object`**

- `boolean equals(Object other)`
- `String toString()`

**`class java.lang.Integer`**

- `Integer(int value)`
- `int intValue()`
- `Integer.MIN_VALUE` // minimum value represented by an `int` or `Integer`
- `Integer.MAX_VALUE` // maximum value represented by an `int` or `Integer`

**`class java.lang.Double`**

- `Double(double value)`
- `double doubleValue()`

**`class java.lang.String`**

- `int length()`
- `String substring(int from, int to)` // returns the substring beginning at `from` // and ending at `to-1`
- `String substring(int from)` // returns `substring(from, length())`
- `int indexOf(String str)` // returns the index of the first occurrence of `str`; // returns `-1` if not found
- `int compareTo(String other)` // returns a value < 0 if `this` is less than `other` // returns a value = 0 if `this` is equal to `other` // returns a value > 0 if `this` is greater than `other`

**`class java.lang.Math`**

- `static int abs(int x)`
- `static double abs(double x)`
- `static double pow(double base, double exponent)`
- `static double sqrt(double x)`
- `static double random()` // returns a `double` in the range [0.0, 1.0)

**`interface java.util.List<E>`**

- `int size()`
- `boolean add(E obj)` // appends `obj` to end of list; returns `true`
- `void add(int index, E obj)` // inserts `obj` at position `index` (0 ≤ `index` ≤ size), // moving elements at position `index` and higher // to the right (adds 1 to their indices) and adjusts size
- `E get(int index)`
- `E set(int index, E obj)` // replaces the element at position `index` with `obj` // returns the element formerly at the specified position
- `E remove(int index)` // removes element from position `index`, moving elements // at position `index + 1` and higher to the left // (subtracts 1 from their indices) and adjusts size // returns the element formerly at the specified position

**`class java.util.ArrayList<E> implements java.util.List<E>`**