Computer Science A: Sample Multiple-Choice Questions

Following is a representative set of questions. The answer key for the Computer Science A multiple-choice questions is on page 43. Multiple-choice scores are based on the number of questions answered correctly. Points are not deducted for incorrect answers, and no points are awarded for unanswered questions. Because points are not deducted for incorrect answers, students are encouraged to answer all multiple-choice questions. Students should eliminate as many choices as they can on any questions for which they do not know the answer, and then select the best answer among the remaining choices.

Directions: Determine the answer to each of the following questions or incomplete statements, using the available space for any necessary scratch work. Then decide which is the best of the choices given and fill in the corresponding circle on the answer sheet. No credit will be given for anything written in the examination booklet. Do not spend too much time on any one problem

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Assume that declarations of variables and methods appear within the context of an enclosing class.
- Assume that method calls that are not prefixed with an object or class name and are not shown within a complete class definition appear within the context of an enclosing class.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.

1. Consider the following code segment.

```
for (int k = 0; k < 20; k = k + 2)
{
  if (k % 3 == 1)
  {
    System.out.print(k + "  ");
  }
}
```

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

(A) `4  16`

(B) `4  10  16`

(C) `0  6  12  18`

(D) `1  4  7  10  13  16  19`

(E) `0  2  4  6  8  10  12  14  16  18`

2. Consider the following code segment.

```
List<String> animals = new ArrayList<String>();

animals.add("dog");
animals.add("cat");
animals.add("snake");
animals.set(2, "lizard");
animals.add(1, "fish");
animals.remove(3);
System.out.println(animals);
```

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

(A) `[dog, fish, cat]`

(B) `[dog, fish, lizard]`

(C) `[dog, lizard, fish]`

(D) `[fish, dog, cat]`

(E) The code throws an `ArrayIndexOutOfBoundsException` exception.

3. Consider the following method.

```
public static void mystery(List<Integer> nums)
{
  for (int k = 0; k < nums.size(); k++)
  {
    if (nums.get(k).intValue() == 0)
    {
      nums.remove(k);
    }
  }
}
```

Assume that a `List<Integer> values` initially contains the following `Integer` values.

```
[0, 0, 4, 2, 5, 0, 3, 0]
```

What will `values` contain as a result of executing `mystery(values)` ?

(A) `[0, 0, 4, 2, 5, 0, 3, 0]`

(B) `[4, 2, 5, 3]`

(C) `[0, 0, 0, 0, 4, 2, 5, 3]`

(D) `[0, 4, 2, 5, 3]`

(E) The code throws an `ArrayIndexOutOfBoundsException` exception.

4. At a certain high school students receive letter grades based on the following scale.

Integer 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 (score >= 84 && score <= 92)
  grade = "B";
if (score >= 75 && score <= 83)
  grade = "C";
if (score < 75)
  grade = "F";
```

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

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

(A) II only

(B) III only

(C) I and II only

(D) I and III only

(E) I, II, and III

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

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

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

6. A car dealership needs a program to store information about the cars for sale. For each car, they want to keep track of the following information: number of doors (2 or 4), whether the car has air conditioning, and its average number of miles per gallon. Which of the following is the best object-oriented program design?

   (A) Use one class, `Car`, with three instance variables:
   `int numDoors`, `boolean hasAir`, and
   `double milesPerGallon`.

   (B) Use four unrelated classes: `Car`, `Doors`, `AirConditioning`, and `MilesPerGallon`.

   (C) Use a class `Car` with three subclasses: `Doors`, `AirConditioning`, and `MilesPerGallon`.

   (D) Use a class `Car`, with a subclass `Doors`, with a subclass `AirConditioning`, with a subclass `MilesPerGallon`.

   (E) Use three classes: `Doors`, `AirConditioning`, and `MilesPerGallon`, each with a subclass `Car`.

7. Consider the following declarations.

```
public interface Shape
{
  int isLargerThan(Shape other);
  // Other methods not shown
}
public class Circle implements Shape
{
  // Other methods not shown
}
```

Which of the following method headings of `isLargerThan` can be added to the declaration of the `Circle` class so that it will satisfy the `Shape` interface?

I. `public int isLargerThan(Shape other)`

II. `public int isLargerThan(Circle other)`

III. `public boolean isLargerThan(Object other)`

(A) I only

(B) II only

(C) III only

(D) I and II only

(E) I, II, and III

Questions 8–9 refer to the following incomplete class declaration.

```
public class TimeRecord
{
  private int hours;
  private int minutes; // 0 ≤ minutes < 60
  /** Constructs a TimeRecord object.
   *  @param h the number of hours
   *           Precondition: h ≥ 0
   *  @param m the number of minutes
   *           Precondition: 0 ≤ m < 60
   */
  public TimeRecord(int h, int m)
  {
    hours = h;
    minutes = m;
  }

  /** @return the number of hours
   */
  public int getHours()
  { /* implementation not shown */ }

  /** @return the number of minutes
   *  Postcondition: 0 ≤ minutes < 60
   */
  public int getMinutes()
  { /* implementation not shown */ }

  /** Adds h hours and m minutes to this TimeRecord.
   *  @param h the number of hours
   *           Precondition: h ≥ 0
   *  @param m the number of minutes
   *           Precondition: m ≥ 0
   */
  public void advance(int h, int m)
  {
    hours = hours + h;
    minutes = minutes + m;
    /* missing code */
  }
  // Other methods not shown
}
```

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

   (A)
   ```
   minutes = minutes % 60;
   ```

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

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

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

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

9. Consider the following declaration that appears in a class other than `TimeRecord`.

   ```
   TimeRecord[] timeCards = new TimeRecord[100];
   ```

   Assume that `timeCards` has been initialized with `TimeRecord` objects. Consider the following code segment that is intended to compute the total of all the times stored in `timeCards`.

   ```
   TimeRecord total = new TimeRecord(0,0);
   for (int k = 0; k < timeCards.length; k++)
   {
     /* missing expression */ ;
   }
   ```

   Which of the following can be used to replace /* *missing expression* */ so that the code segment will work as intended?

   (A)
   ```
   timeCards[k].advance()
   ```

   (B)
   ```
   total += timeCards[k].advance()
   ```

   (C)
   ```
   total.advance(timeCards[k].hours,
                 timeCards[k].minutes)
   ```

   (D)
   ```
   total.advance(timeCards[k].getHours(),
                 timeCards[k].getMinutes())
   ```

   (E)
   ```
   timeCards[k].advance(timeCards[k].getHours(),
                        timeCards[k].getMinutes())
   ```

10. Consider the following instance variable and method.

```
private int[] arr;

/** Precondition: arr contains no duplicates;
 *                the elements in arr are in ascending order.
 *  @param low an int value such that 0 ≤ low ≤ arr.length
 *  @param high an int value such that low - 1 ≤ high < arr.length
 *  @param num an int value
 */
public int mystery(int low, int high, int num)
{
  int mid = (low + high) / 2;
  if (low > high)
  {
    return low;
  }
  else if (arr[mid] < num)
  {
    return mystery(mid + 1, high, num);
  }
  else if (arr[mid] > num)
  {
    return mystery(low, mid - 1, num);
  }
  else // arr[mid] == num
  {
    return mid;
  }
}
```

What is returned by the call `mystery(0, arr.length - 1, num)`?

(A) The number of elements in `arr` that are less than `num`

(B) The number of elements in `arr` that are less than or equal to `num`

(C) The number of elements in `arr` that are equal to `num`

(D) The number of elements in `arr` that are greater than `num`

(E) The index of the middle element in `arr`

Questions 11–12 refer to the following information.

Consider the following instance variable `nums` and method `findLongest` with line numbers added for reference. Method `findLongest` is intended to find the longest consecutive block of the value `target` occurring in the array `nums`; however, `findLongest` does not work as intended.

For example, if the array `nums` contains the values [7, 10, 10, 15, 15, 15, 15, 10, 10, 10, 15, 10, 10], the call `findLongest(10)` should return 3, the length of the longest consecutive block of 10s.

```
private int[] nums;

public int findLongest(int target)
{
    int lenCount = 0;
    int maxLen = 0;

Line 1:  for (int val : nums)
Line 2:  {
Line 3:    if (val == target)
Line 4:    {
Line 5:      lenCount++;
Line 6:    }
Line 7:    else
Line 8:    {
Line 9:      if (lenCount > maxLen)
Line 10:     {
Line 11:       maxLen = lenCount;
Line 12:     }
Line 13:   }
Line 14: }
Line 15: if (lenCount > maxLen)
Line 16: {
Line 17:   maxLen = lenCount;
Line 18: }
Line 19: return maxLen;
    }
```

11. The method `findLongest` does not work as intended. Which of the following best describes the value returned by a call to `findLongest` ?

    (A) It is the length of the shortest consecutive block of the value `target` in `nums`.

    (B) It is the length of the array `nums`.

    (C) It is the number of occurrences of the value `target` in `nums`.

    (D) It is the length of the first consecutive block of the value `target` in `nums`.

    (E) It is the length of the last consecutive block of the value `target` in `nums`.

12. Which of the following changes should be made so that method `findLongest` will work as intended?

    (A) Insert the statement `lenCount = 0;` between lines 2 and 3.

    (B) Insert the statement `lenCount = 0;` between lines 8 and 9.

    (C) Insert the statement `lenCount = 0;` between lines 10 and 11.

    (D) Insert the statement `lenCount = 0;` between lines 11 and 12.

    (E) Insert the statement `lenCount = 0;` between lines 12 and 13.

13. Consider the following instance variable and method.

```
private int[] numbers;

/** Precondition: numbers contains int values in no particular order.
 */
public int mystery(int num)
{
  for (int k = numbers.length - 1; k >= 0; k--)
  {
    if (numbers[k] < num)
    {
      return k;
    }
  }
  return -1;
}
```

Which of the following best describes the contents of `numbers` after the following statement has been executed?

```
int m = mystery(n);
```

(A) All values in positions `0` through `m` are less than `n`.

(B) All values in positions `m+1` through `numbers.length-1` are less than `n`.

(C) All values in positions `m+1` through `numbers.length-1` are greater than or equal to `n`.

(D) The smallest value is at position `m`.

(E) The largest value that is smaller than `n` is at position `m`.

14. Consider the following method.

```
/** @param x an int value such that x >= 0
 */
public void mystery(int x)
{
  System.out.print(x % 10);
  if ((x / 10) != 0)
  {
    mystery(x / 10);
  }
  System.out.print(x % 10);
}
```

Which of the following is printed as a result of the call `mystery(1234)`?

(A) `1234`

(B) `4321`

(C) `12344321`

(D) `43211234`

(E) Many digits are printed due to infinite recursion.

15. Consider the following two classes.

```
public class Dog
{
  public void act()
  {
    System.out.print("run ");
    eat();
  }
  public void eat()
  {
    System.out.print("eat ");
  }
}
public class UnderDog extends Dog
{
  public void act()
  {
    super.act();
    System.out.print("sleep ");
  }
  public void eat()
  {
    super.eat();
    System.out.print("bark ");
  }
 }
```

Assume that the following declaration appears in a class other than `Dog`.

```
Dog fido = new UnderDog();
```

What is printed as a result of the call `fido.act()` ?

(A) `run eat`

(B) `run eat sleep`

(C) `run eat sleep bark`

(D) `run eat bark sleep`

(E) Nothing is printed due to infinite recursion.

16. Consider the following recursive method.

```
public static int mystery(int n)
{
  if (n <= 1)
  {
    return 0;
  }
  else
  {
    return 1 + mystery(n / 2);
  }
}
```

Assuming that k is a nonnegative integer and $m = 2^k$, what value is returned as a result of the call `mystery(`m`)` ?

(A) 0

(B) k

(C) m

(D) $\frac{m}{2} + 1$

(E) $\frac{k}{2} + 1$

17. Consider the following instance variable and method.

```
private int[] array;

/** Precondition: array.length > 0
 */
public int checkArray()
{
  int loc = array.length / 2;
  for (int k = 0; k < array.length; k++)
  {
    if (array[k] > array[loc])
    {
      loc = k;
    }
  }
  return loc;
}
```

Which of the following is the best postcondition for `checkArray` ?

(A) Returns the index of the first element in array `array` whose value is greater than `array[loc]`

(B) Returns the index of the last element in array `array` whose value is greater than `array[loc]`

(C) Returns the largest value in array `array`

(D) Returns the index of the largest value in array `array`

(E) Returns the index of the largest value in the second half of array `array`

18. Consider the following methods.

```
public void changer(String x, int y)
{
  x = x + "peace";
  y = y * 2;
}

public void test()
{
  String s = "world";
  int n = 6;
  changer(s, n);

  /* End of method */
}
```

When the call `test()` is executed, what are the values of `s` and `n` at the point indicated by `/* End of method */` ?

	s	n
(A)	world	6
(B)	worldpeace	6
(C)	world	12
(D)	worldpeace	12
(E)	peace	12

19. Consider the following code segment.

```
int[][] mat = new int[3][4];
for (int row = 0; row < mat.length; row++)
{
   for (int col = 0; col < mat[0].length; col++)
   {
     if (row < col)
     {
       mat[row][col] = 1;
     }
     else if (row == col)
     {
       mat[row][col] = 2;
     }
     else
     {
       mat[row][col] = 3;
     }
   }
}
```

What are the contents of `mat` after the code segment has been executed?

(A)
```
{{2, 1, 1},
 {3, 2, 1},
 {3, 3, 2},
 {3, 3, 3}}
```

(B)
```
{{2, 3, 3},
 {1, 2, 3},
 {1, 1, 2},
 {1, 1, 1}}
```

(C)
```
{{2, 3, 3, 3},
 {1, 2, 3, 3},
 {1, 1, 2, 3}}
```

(D)
```
{{2, 1, 1, 1},
 {3, 2, 1, 1},
 {3, 3, 2, 1}}
```

(E)
```
{{1, 1, 1, 1},
 {2, 2, 2, 2},
 {3, 3, 3, 3}}
```

20. Consider the following method.

```
/** Precondition: arr contains only positive values.
 */
public static void doSome(int[] arr, int lim)
{
  int v = 0;
  int k = 0;
  while (k < arr.length && arr[k] < lim)
  {
    if (arr[k] > v)
    {
      v = arr[k]; /* Statement S */
    }
    k++; /* Statement T */
  }
}
```

Assume that `doSome` is called and executes without error. Which of the following are possible combinations for the value of `lim`, the number of times *Statement S* is executed, and the number of times *Statement T* is executed?

	Value of `lim`	Executions of *Statement S*	Executions of *Statement T*
I.	5	0	5
II.	7	4	9
III.	3	5	2

(A) I only

(B) II only

(C) III only

(D) I and III only

(E) II and III only

21. Consider the following instance variable, `arr`, and incomplete method, `partialSum`. The method is intended to return an integer array `sum` such that for all `k`, `sum[k]` is equal to `arr[0] + arr[1] + ... + arr[k]`. For instance, if `arr` contains the values `{ 1, 4, 1, 3 }`, the array `sum` will contain the values `{ 1, 5, 6, 9 }`.

```
private int[] arr;
public int[] partialSum()
{
  int[] sum = new int[arr.length];
  for (int j = 0; j < sum.length; j++)
  {
    sum[j] = 0;
  }
  /* missing code */
  return sum;
}
```

The following two implementations of `/* missing code */` are proposed so that `partialSum` will work as intended.

Implementation 1

```
for (int j = 0; j < arr.length; j++)
{
  sum[j] = sum[j - 1] + arr[j];
}
```

Implementation 2

```
for (int j = 0; j < arr.length; j++)
{
  for (int k = 0; k <= j; k++)
  {
    sum[j] = sum[j] + arr[k];
  }
}
```

Which of the following statements is true?

(A) Both implementations work as intended, but implementation 1 is faster than implementation 2.

(B) Both implementations work as intended, but implementation 2 is faster than implementation 1.

(C) Both implementations work as intended and are equally fast.

(D) Implementation 1 does not work as intended, because it will cause an `ArrayIndexOutOfBoundsException`.

(E) Implementation 2 does not work as intended, because it will cause an `ArrayIndexOutOfBoundsException`.

22. Consider the following declaration for a class that will be used to represent points in the *xy*-coordinate plane.

```
public class Point
{
  private int x;       // x-coordinate of the point
  private int y;       // y-coordinate of the point

  public Point()
  {
    x = 0;
    y = 0;
  }

  public Point(int a, int b)
  {
    x = a;
    y = b;
  }

  // Other methods not shown
}
```

The following incomplete class declaration is intended to extend the above class so that points can be named.

```
public class NamedPoint extends Point
{
  private String name; // name of point

  // Constructors go here

  // Other methods not shown
}
```

Consider the following proposed constructors for this class.

I.
```
public NamedPoint()
{
  name = "";
}
```

II.
```
public NamedPoint(int d1, int d2, String pointName)
{
  x = d1;
  y = d2;
  name = pointName;
}
```

III.
```
public NamedPoint(int d1, int d2, String pointName)
{
  super(d1, d2);
  name = pointName;
}
```

Which of these constructors would be legal for the `NamedPoint` class?

(A) I only

(B) II only

(C) III only

(D) I and III only

(E) II and III only

23. Consider a `shuffle` method that is intended to return a new array that contains all the elements from `nums`, but in a different order. Let `n` be the number of elements in `nums`. The `shuffle` method should alternate the elements from `nums[0] ... nums[n / 2 - 1]` with the elements from `nums[n / 2] ... nums[n - 1]`, as illustrated in the following examples.

Example 1

	0	1	2	3	4	5	6	7
nums	10	20	30	40	50	60	70	80

	0	1	2	3	4	5	6	7
result	10	50	20	60	30	70	40	80

Example 2

	0	1	2	3	4	5	6
nums	10	20	30	40	50	60	70

	0	1	2	3	4	5	6
result	10	40	20	50	30	60	70

The following implementation of the `shuffle` method does not work as intended.

```
public static int[] shuffle(int[] nums)
{
  int n = nums.length;
  int[] result = new int[n];

  for (int j = 0; j < n / 2; j++)
  {
    result[j * 2] = nums[j];
    result[j * 2 + 1] = nums[j + n / 2];
  }

  return result;
}
```

Which of the following best describes the problem with the given implementation of the `shuffle` method?

(A) Executing `shuffle` may cause an `ArrayIndexOutOfBoundsException`.

(B) The first element of the returned array (`result[0]`) may not have the correct value.

(C) The last element of the returned array (`result[result.length - 1]`) may not have the correct value.

(D) One or more of `nums[0] ... nums[nums.length / 2 - 1]` may have been copied to the wrong position(s) in the returned array.

(E) One or more of `nums[nums.length / 2] ... nums[nums.length - 1]` may have been copied to the wrong position(s) in the returned array.

24. Consider the following `Util` class, which contains two methods. The completed `sum1D` method returns the sum of all the elements of the 1-dimensional array `a`. The incomplete `sum2D` method is intended to return the sum of all the elements of the 2-dimensional array `m`.

```
public class Util
{
  /** Returns the sum of the elements of the 1-dimensional array a */
  public static int sum1D(int[] a)
  {  /* implementation not shown */  }

  /** Returns the sum of the elements of the 2-dimensional array m */
  public static int sum2D(int[][] m)
  {
    int sum = 0;

    /* missing code */

    return sum;
  }
}
```

Assume that `sum1D` works correctly. Which of the following can replace `/* missing code */` so that the `sum2D` method works correctly?

I.
```
for (int k = 0; k < m.length; k++)
{
  sum += sum1D(m[k]);
}
```

II.
```
for (int[] row : m)
{
  sum += sum1D(row);
}
```

III.
```
for (int[] row : m)
{
  for (int v : row)
  {
    sum += v;
  }
}
```

(A) I only

(B) II only

(C) I and II only

(D) II and III only

(E) I, II, and III

25. The following `sort` method correctly sorts the integers in `elements` into ascending order.

```
Line 1:  public static void sort(int[] elements)
Line 2:  {
Line 3:    for (int j = 0; j < elements.length - 1; j++)
Line 4:    {
Line 5.      int index = j;
Line 6:
Line 7:      for (int k = j + 1; k < elements.length; k++)
Line 8:      {
Line 9:        if (elements[k] < elements[index])
Line 10:       {
Line 11:         index = k;
Line 12:       }
Line 13:     }
Line 14:
Line 15:     int temp = elements[j];
Line 16:     elements[j] = elements[index];
Line 17:     elements[index] = temp;
Line 18:   }
Line 19: }
```

Which of the following changes to the `sort` method would correctly sort the integers in `elements` into **descending** order?

I. Replace line 9 with:
Line 9: `if (elements[k] > elements[index])`

II. Replace lines 15–17 with:
Line 15: `int temp = elements[index];`
Line 16: `elements[index] = elements[j];`
Line 17: `elements[j] = temp;`

III. Replace line 3 with:
Line 3: `for (int j = elements.length - 1; j > 0; j--)`
and replace line 7 with:
Line 7: `for (int k = 0; k < j; k++)`

(A) I only

(B) II only

(C) I and II only

(D) I and III only

(E) I, II, and III

AP Computer Science Multiple Choice Practice

Name:

1. _____
2. _____
3. _____
4. _____
5. _____
6. _____
7. _____
8. _____
9. _____
10. _____
11. _____
12. _____
13. _____
14. _____
15. _____
16. _____
17. _____
18. _____
19. _____
20. _____
21. _____
22. _____
23. _____
24. _____
25. _____

APPENDIX B

Exam Appendix — 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` ($0 \le$ `index` $\le$ 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>