Practice Test 3

AP® Computer Science A Exam

SECTION I: Multiple-Choice Questions

DO NOT OPEN THIS BOOKLET UNTIL YOU ARE TOLD TO DO SO.

At a Glance

Total Time
1 hour 30 minutes
Number of Questions
40
Percent of Total Score
50%
Writing Instrument
Pencil required

Instructions

Section I of this examination contains 40 multiple-choice questions. Fill in only the ovals for numbers 1 through 40 on your answer sheet.

Indicate all of your answers to the multiple-choice questions on the answer sheet. No credit will be given for anything written in this exam booklet, but you may use the booklet for notes or scratch work. After you have decided which of the suggested answers is best, completely fill in the corresponding oval on the answer sheet. Give only one answer to each question. If you change an answer, be sure that the previous mark is erased completely. Here is a sample question and answer.

Sample Question

Chicago is a
(A) state
(B) city
(C) country
(D) continent
(E) county

Sample Answer

Use your time effectively, working as quickly as you can without losing accuracy. Do not spend too much time on any one question. Go on to other questions and come back to the ones you have not answered if you have time. It is not expected that everyone will know the answers to all the multiple-choice questions.

About Guessing

Many candidates wonder whether or not to guess the answers to questions about which they are not certain. 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, you are encouraged to answer all multiple-choice questions. On any questions you do not know the answer to, you should eliminate as many choices as you can, and then select the best answer among the remaining choices.

GO ON TO THE NEXT PAGE.

Java Quick Reference

Class Constructors and Methods	Explanation
String Class	
`String(String str)`	Constructs a new `String` object that represents the same sequence of characters as `str`
`int length()`	Returns the number of characters in a `String` object
`String substring(int from, int to)`	Returns the substring beginning at index `from` and ending at index `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
`boolean equals(String other)`	Returns `true` if `this` is equal to `other`; returns `false` otherwise
`int compareTo(String other)`	Returns a value <0 if `this` is less than `other`; returns zero if `this` is equal to `other`; returns a value of >0 if `this` is greater than `other`
Integer Class	
`Integer(int value)`	Constructs a new `Integer` object that represents the specified `int` value
`Integer.MIN_VALUE`	The minimum value represented by an `int` or `Integer`
`Integer.MAX_VALUE`	The maximum value represented by an `int` or `Integer`
`int intValue()`	Returns the value of this `Integer` as an `int`
Double Class	
`Double(double value)`	Constructs a new `Double` object that represents the specified `double` value
`double doubleValue()`	Returns the value of this `Double` as a `double`
Math Class	
`static int abs(int x)`	Returns the absolute value of an `int` value
`static double abs(double x)`	Returns the absolute value of a `double` value
`static double pow(double base, double exponent)`	Returns the value of the first parameter raised to the power of the second parameter
`static double sqrt(double x)`	Returns the positive square root of a `double` value
`static double random()`	Returns a `double` value greater than or equal to `0.0` and less than `1.0`
ArrayList Class	
`int size()`	Returns the number of elements in the list
`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 adds 1 to size
`E get(int index)`	Returns the element at position `index` in the list
`E set(int index, E obj)`	Replaces the element at position `index` with `obj`; returns the element formerly at position `index`
`E remove(int index)`	Removes the element at position `index`, moving elements at position `index + 1` and higher to the left (subtracts `1` from their indices) and subtracts 1 from size; returns the element formerly at position `index`
Object Class	
`boolean equals(Object other)`	
`String toString()`	

GO ON TO THE NEXT PAGE.

COMPUTER SCIENCE A

SECTION I

Time—1 hour and 30 minutes

Number of Questions—40

Percent of total exam grade—50%

Directions: Determine the answer to each of the following questions or incomplete statements, using the available space for any necessary scratchwork. Then decide which is the best of the choices given and fill in the corresponding oval 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 the method calls are not `null` and that methods are called only when their preconditions are satisfied.

1. Consider the following methods.

```
public void trial()
{
      int a = 10;
      int b = 5;
      doubleValues(a, b);
      System.out.print(b);
      System.out.print(a);
}

public void doubleValues(int c, int d)
{
      c = c * 2;
      d = d * 2;
      System.out.print(c);
      System.out.print(d);
}
```

What is printed as the result of the call `trial()`?

(A) `2010`
(B) `2010105`
(C) `2010510`
(D) `20102010`
(E) `20101020`

GO ON TO THE NEXT PAGE.

2. Consider the following method.

```
/**
 * Precondition: a > b > 0
 */
public static int mystery(int a, int b)
{
   int d = 0;
   for (int c = a; c > b; c--)
   {
      d = d + c;
   }
   return d;
}
```

What is returned by the call `mystery(x, y)`?

(A) The sum of all integers greater than `y` but less than or equal to `x`
(B) The sum of all integers greater than or equal to `y` but less than or equal to `x`
(C) The sum of all integers greater than `y` but less than `x`
(D) The sum of all integers greater than or equal to `y` but less than `x`
(E) The sum of all integers less than `y` but greater than or equal to `x`

3. Consider the following method.

```
public void mystery (int  n)
{
   int k;
   for (k = 0 ; k < n ; k++)
   {
      mystery(k);
      System.out.print (k);
   }
}
```

What is printed by the call `mystery(3)` ?

(A) `0123`
(B) `00123`
(C) `0010012`
(D) `00100123`
(E) `001001200100123`

GO ON TO THE NEXT PAGE.

4. Consider an array of integers.

```
4    10    1    2    6    7    3    5
```

If selection sort is used to order the array from smallest to largest values, which of the following represents a possible state of the array at some point during the selection sort process?

(A) `1  4  10  2  3  6  7  5`
(B) `1  2  4  6  10  7  3  5`
(C) `1  2  3  10  6  7  4  5`
(D) `4  3  1  2  6  7  10  5`
(E) `5  3  7  6  2  1  10  4`

5. Consider the following code segment:

```
int  k;
int  a[];
a = new int [7];
for (k = 0; k < a.length; k++)
{
   a[k] = a.length - k;
}
for (k = 0; k < a.length - 1; k++)
{
   a[k+1] = a[k];
}
```

What values will A contain after the code segment is executed?

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

GO ON TO THE NEXT PAGE.

Questions 6–7 refer to the following two classes.

```
public class PostOffice
{
    // constructor initializes boxes
    // to length 100
    public PostOffice()
    {   /* implementation not shown   */}

    // returns the P.O. Box based on the given P.O. Box number
    //  0 <=  theBox  <  getNumBoxes ()
    public Box getBox(int theBox)
    {   /* implementation not shown   */}
    // returns the number of p.o. boxes
    public int getNumBoxes()
    {   /* implementation not shown   */}

    // private data members and
    // other methods not shown
}

public class Box
{
    // constructor
    public Box()
    {   /* implementation not shown  */}

    // returns the number of this box
    public int getBoxNumber()
    {   /* implementation not shown   */}

    // returns the number of pieces
    // of mail in this box
    public int getMailCount()
    {   /* implementation not shown  */}
    // returns the given piece of mail
    // 0 <= thePiece < getMailCount ( )
    public Mail getMail(int thePiece)
    {   /* implementation not shown  */}
    // true if the box has been assigned
    // to a customer
    public boolean isAssigned()
    {   /* implementation not shown  */}
    // true if the box contains mail
    public boolean hasMail()
    {   /* implementation not shown  */}
    // private data members and
    // other methods not shown
}
public class Mail
{
    // private members,  constructors, and
    // other methods not shown
}
```

GO ON TO THE NEXT PAGE.

6. Consider the following code segment:

```
PostOffice p[ ];
p = new PostOffice[10];
```

Assuming that the box has been assigned and that it has at least four pieces of mail waiting in it, what is the correct way of getting the fourth piece of mail from the 57th box of the 10th post office of `p`?

(A) `Mail m = p[10].getBox(57).getmail(4);`
(B) `Mail m = p[9].getBox(56).getMail(3);`
(C) `Mail m = p.getMail(57).getMail(4) [10];`
(D) `Mail m = getMail(getBox(p[9], 560, 3);`
(E) `Mail m = new Mail(10, 57, 4);`

GO ON TO THE NEXT PAGE.

7. Consider the incomplete function `printEmptyBoxes` given below. `printEmptyBoxes` should print the box numbers of all of the boxes that have been assigned to a customer but do not contain mail.

```
public void printEmptyBoxes (PostOffice[] p)
{
   for (int k = 0; k < p.length - 1 ; k++)
   {
      for (int x = 0; x < p[k].getNumBoxes() - 1 ; x++)
      {
           /* missing code */
      }
   }
}
```

Which of the following could be used to replace `/* missing code */` so that `printBoxesWithoutMail` works as intended?

(A)
```
if (p[k].getBox(x).isAssigned() &&
!p[k].getBox(x).hasMail())
{
     System.out.println(p[k].getBox(x).getBoxNumber()) ;
}
```

(B)
```
if (p[x].getBox(k).isAssigned() &&
!p[x].getBox(k).hasMail())
{
  System.out.println(p[x].getBox(k).getBoxNumber());
}
```

(C)
```
if (p[k].getBox(x).isAssigned() &&
!p[k].getBox(x).hasMail())
{
     System.out.println (p[k].getBoxNumber (x));
}
```

(D)
```
if (p[x].getBox(k).isAssigned() &&
!p[x].getBox (k).hasMail())
{
     System.out.println(p[x].getBoxNumber(k));
  }
```

(E)
```
if (p[x].getBox(k).isAssigned() &&
p[x].getBox(k).getMail() == 0)
{
     System.out.println(k);
}
```

GO ON TO THE NEXT PAGE.

8. Assume that a and b are boolean variables that have been initialized. Consider the following code segment.

```
a = a && b;
b = a || b;
```

Which of the following statements is always true?

I. The final value of a is the same as the initial value of a.
II. The final value of b is the same as the initial value of b.
III. The final value of a is the same as the initial value of b.

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

9. Consider the following code segment.

```
int x;
x = 53;
if (x > 10)
{
   System.out.print("A");
}
if (x > 30)
{
   System.out.print("B");
}
else if (x > 40)
{
   System.out.print("C");
}
if (x > 50)
{
   System.out.print ("D");
}
if (x > 70)
{
   System.out.print ("E");
}
```

What is the output when the code is executed?

(A) A
(B) D
(C) ABD
(D) ABCD
(E) ABCDE

GO ON TO THE NEXT PAGE.

10. Consider the following code segment:

```
int j;
int k;
for (j = -2; j <= 2; j = j + 2)
{
   for (k = j; k < j + 3; k++)
   {
      System.out.print(k + " ");
   }
}
```

What is the output when the code is executed?

(A) -2 -1 0
(B) -2 -1 0 1 2
(C) 0 1 2 0 1 2 0 1 2
(D) -2 0 2
(E) -2 -1 0 0 1 2 2 3 4

11. Consider the following method.

```
public void mystery (int  count, String s)
{
   if  (count  <=  0)
   {
      return;
   }
   if  (count % 3 == 0)
   {
      System.out.print(s + "--" + s);
   }
   else if (count % 3 == 1)
   {
      System.out.print(s + "-" + s);
   }
   else
   {
      System.out.print(s);
   }
   mystery(count - 1, s);
}
```

What is outputted by the call `mystery(5, "X")`?

(A) XX-XX--XXX-X
(B) XX-XX-XX-XX
(C) XXX--XX-X-XX--XXX
(D) XX-XXX--XXX-XX
(E) XXXXX

GO ON TO THE NEXT PAGE.

Questions 12–13 refer to the following classes and method descriptions.

Class `Table` has a method, `getPrice`, which takes no parameters and returns the price of the table.

Class `Chair` also has a method, `getPrice`, which takes no parameters and returns the price of the chair.

Class `DiningRoomSet` has a constructor which is passed a `Table` object and an `ArrayList` of `Chair` objects. It stores these parameters in its private data fields `myTable` and `myChairs`.

Class `DiningRoomSet` has a method, `getPrice`, which takes no parameters and returns the price of the dining room set. The price of a dining room set is calculated as the sum of the price of its table and all of its chairs.

12. What is the correct way to define the signature of the constructor for the `DiningRoomSet` class?

(A) `public void DiningRoomSet(Table t, ArrayList, chairs)`
(B) `public DiningRoomSet(Table t, ArrayList<Chair> chairs)`
(C) `public void DiningRoomSet(Table t, ArrayList Chair Chairs)`
(D) `public DiningRoomSet(Table t, ArrayList Chair Chairs)`
(E) `public DiningRoomSet(Table t, Chair Chairs)`

13. What is the correct way to implement the getPrice method of the `DiningRoomSet` class?

(A)
```
public double getPrice(Table t, ArrayList chairs)
{
     return t.getPrice() + chairs.getPrice();
}
```
(B)
```
public double getPrice(Table t, ArrayList chairs)
{
     return myTable.getPrice() + myChairs.getPrice();
}
```
(C)
```
public double getPrice()
{
     return myTable.getPrice() + myChairs.getPrice();
}
```
(D)
```
public double getPrice()
{
     double result  = myTable.getPrice();
     for (int k = 0; k < myChairs.size() - 1; k++)
     {
          result += ((Chair)myChairs.get(k)).getPrice();
     }
     return result;
}
```
(E)
```
public double getPrice()
{
      double result = myTable.getPrice();
      for (int k = 0; k  <  myChairs.length - 1; k++)
      {
           result += ((Chair)myChairs[k]).getPrice();
      }
      return result;
}
```

GO ON TO THE NEXT PAGE.

14. Consider the following output:

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

Which of the following code segments produces the above output when executed?

(A)
```
for  (int j = 6;  j  <  0;  j--)
{
  for (int k = j; k > 0; k--)
  {
      System.out.print(k  +  "   ");
  }
  System.out.println(" ");
}
```

(B)
```
for (int j = 6; j >= 0; j--)
{
  for (int k = j; k >= 0; k--)
  {
      System.out.print(k + " ");
  }
  System.out.println(" ");
}
```

(C)
```
for (int j = 0; j < 6; j++)
{
  for (int k = 6 - j; k > 0; k--)
  {
      System.out.print(k + "  ");
  }
  system.out.println(" ");
}
```

(D)
```
for (int j = 0; j < 6; j++)
{
  for (int k = 7 - j ; k > 0 ; k--)
  {
      System.out.print(k + " ");
  }
  System.out.println(" ");
}
```

(E)
```
for (int j = 0; j < 6; j++)
{
  for (int k = 6  -  j ; k >= 0; k--)
  {
      System.out.print(k + " ");
  }
  System.out.println(" ");
}
```

GO ON TO THE NEXT PAGE.

15. Consider the following code segment.

```
List<Integer> list = new ArrayList<Integer>();
list.add(new Integer(7));
list.add(new Integer(6));
list.add(1, new Integer(5));
list.add(1, new Integer(4));
list.add(new Integer(3));
list.set(2, new Integer(2));
list.add(1, new Integer(1));
System.out.println(list);
```

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

(A) `[1, 4, 2, 7, 6, 3]`
(B) `[7, 1, 4, 2, 6, 3]`
(C) `[7, 2, 5, 4, 3, 1]`
(D) `[7, 6, 2, 4, 3, 1]`
(E) `[7, 1, 2]`

16. Consider the following declarations.

```
public class Animal
{
    String makeSound()
    {
       // Implementation not shown
    }
    String animalType()
    {
       // Implementation not shown
    }
}
public static class Dog extends Animal
{
    public String makeSound(Animal a)
    {
       // Implementation not shown
    }
}
```

Which of the following methods must be included in the declaration of the `Dog` class in order for the class to successfully compile?

I. `public String makeSound()`
II. `public String animalType()`
III. `public String animalType(Animal b)`

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

17. Consider the following two classes.

```
public class Fish
{
  public String endoskeleton = “bone”;

  public void action()
  {
      System.out.println(“splash splash”);
  }
}

public class Shark extends Fish
{
  public void action()
  {
      System.out.println(“chomp chomp”);
  }

  public String endoskeleton = “cartilage”;
}
```

Which of the following is the correct output after the following code segment is executed?

```
Fish Bob = new Shark();
System.out.println(Bob.endoskeleton);
Bob.action();
```

(A) bone
chomp chomp

(B) bone
splash splash

(C) cartilage
splash splash

(D) cartilage
chomp chomp

(E) cartilage
splash splash
chomp chomp

GO ON TO THE NEXT PAGE.

Questions 18–19 refer to the following incomplete method.

The following `insertSort` method sorts the values in an integer array, `sort`, in ascending order.

```
1   public static void insertSort(int[] sort)
2      {
3         for (int index = 1;index < sort.length;index++)
4         {
5         int temp = sort[index];
6         while (index > 0 && sort[index - 1] > temp)
7               {
8         /* missing code */
9               }
10     sort[index] = temp;
11        }
12     }
```

18. Which of the following can be used to replace `/* missing code */` so that the `insertSort` method will execute properly?

(A)
```
sort[index] = sort[index - 1];
index++;
```
(B)
```
sort[index - 1] = sort[index];
index--;
```
(C)
```
sort[index] = sort[index + 1];
index++;
```
(D)
```
sort[index] = sort[index - 1];
index--;
```
(E)
```
sort[index] = sort[index + 1];
index--;
```

19. Assuming that the `/* missing code */` is implemented properly, what change can be made to the code in order for the array to be sorted in descending order?

(A) Replace Line 6 with: `while (index < 0 && sort[index - 1] > temp)`
(B) Replace Line 6 with: `while (index < 0 && sort[index - 1] < temp)`
(C) Replace Line 6 with: `while (index > 0 && sort[index - 1] < temp)`
(D) Replace Line 3 with: `for (int index = sort.length - 1;index > 0;index--)`
(E) Replace Line 3 with: `for (int index = 1;index > 0;index--)`

20. Which of the following arrays would be sorted the slowest using insertion sort?

(A) `[3 4 6 2 7 3 9]`
(B) `[3 2 5 4 6 7 9]`
(C) `[9 7 6 5 4 3 2]`
(D) `[2 3 4 5 6 7 9]`
(E) `[9 3 2 4 5 7 6]`

GO ON TO THE NEXT PAGE.

Questions 21–23 refer to the following incomplete class declaration used to represent fractions with integer numerators and denominators.

```
public class Fraction
{
  private int numerator;
  private int denominator;

  public Fraction()
  {
      numerator = 0;
      denominator = 1;
  }

  public Fraction(int n, int d)
  {
      numerator = n;
      denominator = d;
  }

  // postcondition: returns the
  //     numerator
  public int getNumerator()
  {   /* implementation not shown */  }

  // postcondition: returns the
  //     denominator
  public int getDenominator()
  {   /* implementation not shown*/   }

  // postcondition: returns the greatest
  // common divisor of x and y
  public int gcd(int x, int y)
  {   /* implementation not shown*/   }

  // postcondition: returns the Fraction
  //     that is the result of multiplying
  //     this Fraction and f

  public Fraction multiply(Fraction f)
  {   /* implementation not shown */  }
  // . . . other methods not shown
}
```

GO ON TO THE NEXT PAGE.

21. Consider the method `multiply` of the `Fraction` class.

```
// postcondition: returns the Fraction
//          that is the result of multiplying
//          this Fraction and f
public Fraction multiply(Fraction f)
{  /* missing code */  }
```

Which of the following statements can be used to replace `/* missing code */` so that the `multiply` method is correctly implemented?

I.
```
return Fraction(
  numerator * f.getNumerator(),
  denominator * f.getDenominator());
```

II.
```
return new Fraction(
  numerator * f.numerator,
  denominator * f.denominator());
```

III.
```
return new Fraction(
  numerator * f.getNumerator(),
  denominator * f.getDenominator());
```

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

22. Consider the use of the `Fraction` class to multiply the fractions $\frac{3}{4}$ and $\frac{7}{19}$. Consider the following code:

```
Fraction fractionOne;
Fraction fractionTwo;
Fraction answer;
fractionOne = new Fraction(3, 4);
fractionTwo = new Fraction(7, 19);
/* missing code */
```

Which of the following could be used to replace `/* missing code */` so that the answer contains the result of multiplying fractionOne by fractionTwo?

(A)
```
answer = fractionOne * fractionTwo
```
(B)
```
answer = multiply(fractionOne,fractionTwo);
```
(C)
```
answer = fractionOne.multiply(fractionTwo);
```
(D)
```
answer = new Fraction(fractionOne,fractionTwo);
```
(E)
```
answer = (fractionOne.getNumerator() * fractionTwo.getNumerator()) /
      (fractionOne.getDenominator() * fractionTwo.getDenominator());
```

GO ON TO THE NEXT PAGE.

23. The following incomplete class declaration is intended to extend the `Fraction` class so that fractions can be manipulated in reduced form (lowest terms).

Note that a fraction can be reduced to lowest terms by dividing both the numerator and denominator by the greatest common divisor (gcd) of the numerator and denominator.

```
public class ReducedFraction extends Fraction
{
   private int reducedNumerator;
   private int reducedDenominator;
   // . . . constructors and other methods not shown
}
```

Consider the following proposed constructors for the `ReducedFraction` class:

I.
```
public ReducedFraction( )
{
  reducedNumerator = 0;
  reducedDenominator = 1;
}
```

II.
```
public ReducedFraction(int n, int d)
{
  numerator = n;
  denominator = d;
  reducedNumerator = n / gcd(n, d);
  reducedDenominator = d / gcd(n, d);
}
```

III.
```
public ReducedFraction(int n, int d)
{
  super(n, d);
  reducedNumerator = n / gcd(n, d);
  reducedDenominator = d / gcd(n, d);
}
```

Which of these constructor(s) would be legal for the `ReducedFraction` class?

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

GO ON TO THE NEXT PAGE.

24. Consider `s1` and `s2` defined as follows.

```
String s1 = new String("hello");
String s2 = new String("hello");
```

Which of the following is/are correct ways to see if `s1` and `s2` hold identical strings?

I.
```
if (s1 == s2)
  /* s1 and s2 are identical */
```
II.
```
if (s1.equals(s2))
  /* s1 and s2 are identical */
```
III.
```
if (s1.compareTo(s2) == 0)
  /* s1 and s2 are identical */
```

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

25. Consider the following variable and method declarations:

```
String s;
String t;
public void mystery (String a, String b)
{
   a = a + b;
   b = b + a;
}
```

Assume that `s` has the value "`Elizabeth`" and `t` has the value "`Andrew`" and `mystery(s, t)` is called. What are the values of `s` and `t` after the call to `mystery`?

	`s`	`t`
(A)	Elizabeth	Andrew
(B)	ElizabethAndrew	AndrewElizabeth
(C)	ElizabethAndrew	AndrewElizabethAndrew
(D)	ElizabethAndrew	ElizabethAndrewAndrew
(E)	ElizabethAndrewElizabeth	AndrewElizabethAndrew

GO ON TO THE NEXT PAGE.

26. Consider the following incomplete and *incorrect* class and interface declarations:

```
public class Comparable Object
{
   public int compareTo(Object o)
   {
      //method body not shown
   }
//other methods and variables not shown
}
public class Point extends ComparableObject
{
   private int x;
   private int y;
   public boolean compareTo(Point other)
   {
      return (x == other.x &&
              y == other.y);
   }
   // . . . constructors and other methods
   //    not shown
}
```

For which of the following reasons is the above class structure incorrect?

I. Objects may not access private data fields of other objects in the same class.
II. The `ComparableObject` class requires that `compareTo` be passed as an `Object` rather than a `Point`.
III. The `ComparableObject` class requires that `compareTo` return an `int` rather than a `boolean`.

(A) I only
(B) III only
(C) I and III only
(D) II and III only
(E) None, the above class declarations are correct.

GO ON TO THE NEXT PAGE.

27. Consider the following abstraction of a `for` loop where `<1>, <2>, <3>,` and `<4>` represent legal code in the indicated locations:

```
for (<1>; <2>; <3>)
{
   <4>
}
```

Which of the following `while` loops has the same functionality as the above `for` loop?

(A)
```
<1>;
while (<2>)
{
   <3>;
   <4>
}
```

(B)
```
<1>;
while (<2>)
{
   <4>
   <3>;
}
```

(C)
```
<1>;
   while (!<2>)
{
   <3>;
   <4>
}
```

(D)
```
<1>;
while (!<2>)
{
   <4>
   <3>;
}
```

(E)
```
<1>;
<3>;
while (<2>)
{
   <4>
   <3>;
}
```

28. Consider the following expression:

```
a / b + c - d % e * f
```

Which of the expressions given below is equivalent to the one given above?

(A) `((a / b) + (c - d)) % (e * f)`
(B) `((((a / b) + c) - d) % e) * f`
(C) `((a / b) + c) - (d % (e * f)`
(D) `(a / ((b + c) - d) % e) * f`
(E) `((a / b) + c) - ((d % e) * f)`

GO ON TO THE NEXT PAGE.

29. Assume that a program declares and initializes x as follows:

```
String[] x ;
x = new String[10] ;
initialize(x);          // Fills the array x with
                        // valid strings each of
                        // length 5
```

Which of the following code segments correctly traverses the array and prints out the first character of all ten strings followed by the second character of all ten strings, and so on?

I.
```
int i;
int j;
for (i = 0 ; i < 10 ; i++)
     for (j = 0 ; j < 5 ; j++)
         System.out.print(x[i].substring(j, j + 1));
```

II.
```
int i;
int j;
for (i = 0 ; i < 5 ; i++)
    for (j = 0 ; j < 10 ; j++)
        System.out.print(x[j].substring(i, i + 1));
```

III.
```
int i ;
int j ;
for (i = 0 ; i < 5 ; i++)
    for (j = 0 ; j < 10 ; j++)
        System.out.print(x[i].substring(j, j + 1));
```

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

GO ON TO THE NEXT PAGE.

30. Consider the following declaration and assignment statements:

```
int a = 7;
int b = 4;
double c;
c = a / b;
```

After the assignment statement is executed, what's the value of `c`?

(A) 1.0
(B) 1.75
(C) 2.0
(D) An error occurs because `c` was not initialized.
(E) An error occurs because `a` and `b` are integers and `c` is a double.

31. Consider the following code segment:

```
int x;
x = /* initialized to an integer */
if (x % 2 == 0 && x / 3 == 1)
   System.out.print("Yes");
```

For what values of `x` will the word "Yes" be printed when the code segment is executed?

(A) 0
(B) 4
(C) Whenever `x` is even and `x` is not divisible by 3
(D) Whenever `x` is odd and `x` is divisible by 3
(E) Whenever `x` is even and `x` is divisible by 3

GO ON TO THE NEXT PAGE.

32. Consider the following incomplete class definition:

```
public class SomeClass
{
   private String myName;
   // postcondition: returns myName
   public String getName( )
   { /* implementation not shown */ }
   // postcondition: myName == name
   public void setName(String name)
   { /* implementation not shown */ }
   // . . . constructors, other methods
   //    and private data not shown
}
```

Now consider the method `swap`, not part of the `SomeClass` class.

```
// precondition: x and y are correctly
//    constructed
// postcondition: the names of objects
//    x and y are swapped
public void swap (SomeClass x, SomeClass y)
{
   /* missing code */
}
```

Which of the following code segments can replace */* missing code */* so that the method `swap` works as intended?

I.
```
SomeClass temp;
temp = x;
x = y;
y = temp;
```

II.
```
String temp;
temp = x.myName;
x.myName = y.myName;
y.myName = temp;
```

III.
```
String temp;
temp = x.getName();
x.setName(y.getName());
y.setName(temp);
```

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

GO ON TO THE NEXT PAGE.

33. A bookstore wants to store information about the different types of books it sells.

For each book, it wants to keep track of the title of the book, the author of the book, and whether the book is a work of fiction or nonfiction.

If the book is a work of fiction, then the bookstore wants to keep track of whether it is a romance novel, a mystery novel, or science fiction.

If the book is a work of nonfiction, then the bookstore wants to keep track of whether it is a biography, a cookbook, or a self-help book.

Which of the following is the best design?

(A) Use one class, `Book`, which has three data fields: `String title`, `String author`, and `int bookType`.
(B) Use four unrelated classes: `Book`, `Title`, `Author`, and `BookType`.
(C) Use a class `Book` which has two data fields: `String title`, `String author`, and a subclass: `BookType`.
(D) Use a class `Book` which has two data fields: `String title`, `String author`, and six subclasses: `RomanceNovel`, `Mystery`, `ScienceFiction`, `Biography`, `Cookbook`, and `SelfHelpBook`.
(E) Use a class `Book` which has two data fields: `String title`, `String author`, and two subclasses: `FictionWork` and `NonFictionWork`. The class `FictionWork` has three subclasses, `RomanceNovel`, `Mystery`, and `ScienceFiction`. The class `NonFictionWork` has three subclasses: `Biography`, `Cookbook`, and `SelfHelpBook`.

GO ON TO THE NEXT PAGE.

34. Consider the following code:

```
public int mystery(int x)
{
   if (x == 1)
      return <missing value>;
   else
      return(2 * mystery(x - 1)) + x;
}
```

Which of the following can be used to replace `<missing value>` so that `mystery (4)` returns `34` ?

(A) 0
(B) 1
(C) 2
(D) 3
(E) 4

35. Consider the following code segment:

```
int [ ] X;
int [ ] Y;
int k;
X = initializeX();       // returns a valid
                         // initialized int [ ]
Y = initializeY();       // returns a valid
                         // initialized int [ ]
for (k = 0;
   k < X.length && X[k] == Y[k];
   k++)
{
   /* some code */
}
```

Assuming that after `X` and `Y` are initialized, `X.length == Y.length`, which of the following must be true after executing this code segment?

(A) `k < X.length`
(B) `k < X.length && X[k] == Y[k]`
(C) `k < X.length && X[k] != Y[k]`
(D) `k >= X.length || X[k] == Y[k]`
(E) `k >= X.length || X[k] != Y[k]`

36. Which of the following would NOT cause a run-time exception?

(A) Dividing an integer by zero
(B) Using an object that has been declared but not instantiated
(C) Accessing an array element with an array index that is equal to the length of the array
(D) Attempting to create a substring beginning at a negative index
(E) Attempting to call a method with the wrong number of arguments

GO ON TO THE NEXT PAGE.

37. Assume that a and b are properly initialized variables of type `Double`.

Which of the following is an equivalent expression to:

```
a.doubleValue()   != b.doubleValue()
```

(A) `a != b`
(B) `a.notEquals(b)`
(C) `!(a.doubleValue() .equals(b.doubleValue()))`
(D) `!(a.compareTo(b))`
(E) `a.compareTo(b) != 0`

38. Which of the following would be the LEAST effective way of ensuring reliability in a program?

(A) Encapsulating functionality in a class by declaring all data fields to be public
(B) Defining and following preconditions and postconditions for every method
(C) Including assertions at key places in the code
(D) Using descriptive variable names
(E) Indenting code in a consistent and logical manner

39. Consider a dictionary that has 1,024 pages with 50 words on each page.

In order to look up a given target word, a student is considering using one of the following three methods:

Method 1

Use a binary search technique to find the correct page (comparing the target word with the first word on a given page). When the correct page is found, use a sequential search technique to find the target word on the page.

Method 2

Use a sequential search technique to find the correct page (comparing the target word with the first word on a given page). When the correct page is found, use another sequential search technique to find the target word on the page.

Method 3

Use a sequential search technique on all of the words in the dictionary to find the target word.

Which of the following best characterizes the greatest number of words that will be examined using each method?

	Method 1	Method 2	Method 3
(A)	10	50	1,024
(B)	55	512	2,560
(C)	55	537	25,600
(D)	60	1,074	1,074
(E)	60	1,074	51,200

GO ON TO THE NEXT PAGE.

40. Consider the following recursive method.

```
public static int mystery(int m)
{
   if (m == 0)
   {
      return 0;
   }
   else
   {
      return 4 + mystery(m - 2);
   }
}
```

Assuming that j is a positive integer and that m = 2j, what value is returned as a result of the call `mystery(m)`?

(A) `0`
(B) `m`
(C) `2m`
(D) `j`
(E) `2j`

END OF SECTION I

IF YOU FINISH BEFORE TIME IS CALLED,
YOU MAY CHECK YOUR WORK ON THIS SECTION.

DO NOT GO ON TO SECTION II UNTIL YOU ARE TOLD TO DO SO.

GO ON TO THE NEXT PAGE.

COMPUTER SCIENCE A

SECTION II

Time—1 hour and 30 minutes

Number of Questions—4

Percent of Total Grade—50%

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA™.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- 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.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

FREE-RESPONSE QUESTIONS

1. A day care has a program to keep track of its employees and which children they teach during the day. An `Employee` has a minimum and maximum age they can teach. The `DayCare` also has a maximum ratio that specifies the maximum number of children a single employee can teach. Below is the full `DayCare` class:

```
public class DayCare
{
    private ArrayList<Employee> employees;
    private ArrayList<Child> children;
    private int maxRatio;

    public DayCare(int maxRatio)
    {
       employees = new ArrayList<Employee>();
       children = new ArrayList<Child>();
       this.maxRatio = maxRatio;
    }

    public boolean findEmployeeForChild(Child c)
    {
       /* To be completed in part (a) */
    }

    public boolean runDayCare()
    {
       /* To be completed in part (b) */
    }

    public boolean addChild(Child c)
    {
       /* To be completed in part (c) */
    }
}
```

GO ON TO THE NEXT PAGE.

(a) An `Employee` can only teach children between the employee's minimum age (inclusive) and maximum age (inclusive). They can also only teach children up to the day care's maximum ratio (inclusive). Below is the full `Employee` class.

```
public class Employee
{
    /* Instance variables not shown */

    public Employee(String name, String id, int min, int max)
    {
        /* Implementation not shown */
    }

    // Return the number of children currently assigned to this Employee
    public int childrenAssigned()
    {
        /* Implementation not shown */
    }

    // Assign a new child to this Employee
    public void assignChild(Child c)
    {
        /* Implementation not shown */
    }

    // Determine whether this Employee can teach a Child based on the child's age
    public boolean canTeach(int age)
    {
        /* Implementation not shown */
    }
}
```

A `Child` has accessors to get their name and age. While the implementation of `Child` is not shown, you can assume the accessors are called `getName` and `getAge`.

Complete the `findEmployeeForChild` method below that assigns a `Child` to the first `Employee` who can teach the `Child` and who has not reached the maximum ratio of the `DayCare`.

```
/* Return true if an Employee was found for the Child, false otherwise */
public boolean findEmployeeForChild(Child c)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

(b) In order for the `DayCare` to run for a day, each `Child` must be assigned an `Employee`. If an `Employee` cannot be found for a `Child`, the `DayCare` cannot run for the day.

Complete the `runDayCare` method below that finds an `Employee` for each `Child` in the `children ArrayList`.

```
/* Return true if an Employee was found for each Child, false otherwise */
public boolean runDayCare()
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

(c) When a `Child` is added to the roster of the `DayCare`, the `DayCare` should first make sure there is an `Employee` available to teach that `Child`.

Complete the `addChild` method below that adds a `Child` to the `children ArrayList` if an `Employee` is available to teach that `Child`.

```
/* Return true if the Child was added to the ArrayList, false otherwise */
public boolean addChild(Child c)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

2. A baseball team consists of different people including players, coaches, and people who work in the front office making trades and other transactions. The following `Person` class is used for all of the people who work for the team.

Each person has a name and an age.

```
public class Person
{
    private String fullName;
    private int age;

    public Person(String s, int a)
    {
       fullName = s;
       age = a;
    }

    // Accessors for name and age
    public String getName()
    {
       return fullName;
    }

    public int getAge()
    {
       return age;
    }
}
```

GO ON TO THE NEXT PAGE.

A `Player` has a name and age just like any person on the team, but also has a `position`. The position could be something like "catcher," "left fielder," or "infielder." Players should also be able to change their positions using a method called `changePosition`. Here is an example of a `Player` object:

```
Player p = new Player("Sammy Sosa", 32, "right fielder");
p.changePosition("outfielder");
```

Write the entire `Player` class.

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

3. A class is designed to store someone's full name. You can assume the name has only one space between the first name and the last name. The class has methods to extract the first name, last name, and number of vowels in the name. You can see an example below.

```
String fullName = "Katherine Johnson";
Name.getFirstName(fullName); // Returns "Katherine"
Name.getLastName(fullName); // Returns "Johnson"
Name.countVowels(fullName); // Returns 6
```

(a) The `getFirstName` method returns the first name based on a given full name. You can assume that `fullName` has only one space between the first name and the last name. Write the `getFirstName` method.

```
public static String getFirstName(String name)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

(b) The `getLastName` method returns the last name based on a given full name. You can assume that `fullName` has only one space between the first name and the last name. Write the `getLastName` method.

```
public static String getLastName(String name)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

(c) The `countVowels` method counts the number of vowels in the given full name. You can assume we will count only the letters `a`, `e`, `i`, `o`, and `u` as vowels. Write the entire `countVowels` method.

```
public static int countVowels(String name)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

4. A city parking lot has a sign that keeps track of how many parking spaces are available in the lot. The class for the parking lot is detailed below.

```
public class ParkingLot
{
    private Car[ ][ ] lot;

    public ParkingLot(int rows, int cols)
    {
       lot = new Car[rows][cols];
    }

    public int openSpaces()
    {
       // Complete in part (a)
    }

    public boolean parkCar(Car newCar)
    {
       // Complete in part (b)
    }
}
```

GO ON TO THE NEXT PAGE.

(a) Write the `openSpaces` method that returns the number of spaces available in the parking lot. If a space is empty, it will be equal to `null`.

```
/* Return the number of empty spaces in the parking lot */
public int openSpaces( )
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

(b) Complete the `parkCar` method that puts a new car in any space in the parking lot and returns `true` if it was able to do so. It should return `false` if there are no empty spaces. You should use the `openSpaces` method to receive full credit.

```
/* Return true if there is an open spot to park the newCar, false otherwise. The car should be added to the lot 2D array
if there is an open spot. */
public boolean parkCar(Car newCar)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

STOP

END OF EXAM

The Princeton Review®

Completely darken bubbles with a No. 2 pencil. If you make a mistake, be sure to erase mark completely. Erase all stray marks.

1.

YOUR NAME: (Print) _____ Last _____ First _____ M.I.

SIGNATURE: _____ DATE: ___/___/___

HOME ADDRESS: (Print) _____ Number and Street

_____ City _____ State _____ Zip Code

PHONE NO.: _____

IMPORTANT: Please fill in these boxes exactly as shown on the back cover of your test book.

2. TEST FORM

3. TEST CODE

4. REGISTRATION NUMBER

0	A	J	0	0	0	0	0	0	0	0	0
1	B	K	1	1	1	1	1	1	1	1	1
2	C	L	2	2	2	2	2	2	2	2	2
3	D	M	3	3	3	3	3	3	3	3	3
4	E	N	4	4	4	4	4	4	4	4	4
5	F	O	5	5	5	5	5	5	5	5	5
6	G	P	6	6	6	6	6	6	6	6	6
7	H	Q	7	7	7	7	7	7	7	7	7
8	I	R	8	8	8	8	8	8	8	8	8
9			9	9	9	9	9	9	9	9	9

5. YOUR NAME

First 4 letters of last name					FIRST INIT	MID INIT
A	A	A	A		A	A
B	B	B	B		B	B
C	C	C	C		C	C
D	D	D	D		D	D
E	E	E	E		E	E
F	F	F	F		F	F
G	G	G	G		G	G
H	H	H	H		H	H
I	I	I	I		I	I
J	J	J	J		J	J
K	K	K	K		K	K
L	L	L	L		L	L
M	M	M	M		M	M
N	N	N	N		N	N
O	O	O	O		O	O
P	P	P	P		P	P
Q	Q	Q	Q		Q	Q
R	R	R	R		R	R
S	S	S	S		S	S
T	T	T	T		T	T
U	U	U	U		U	U
V	V	V	V		V	V
W	W	W	W		W	W
X	X	X	X		X	X
Y	Y	Y	Y		Y	Y
Z	Z	Z	Z		Z	Z

6. DATE OF BIRTH

Month	Day		Year	
JAN				
FEB	0	0	0	0
MAR	1	1	1	1
APR	2	2	2	2
MAY	3	3	3	3
JUN		4	4	4
JUL		5	5	5
AUG		6	6	6
SEP		7	7	7
OCT		8	8	8
NOV		9	9	9
DEC				

The Princeton Review®

1. A B C D E	11. A B C D E	21. A B C D E	31. A B C D E
2. A B C D E	12. A B C D E	22. A B C D E	32. A B C D E
3. A B C D E	13. A B C D E	23. A B C D E	33. A B C D E
4. A B C D E	14. A B C D E	24. A B C D E	34. A B C D E
5. A B C D E	15. A B C D E	25. A B C D E	35. A B C D E
6. A B C D E	16. A B C D E	26. A B C D E	36. A B C D E
7. A B C D E	17. A B C D E	27. A B C D E	37. A B C D E
8. A B C D E	18. A B C D E	28. A B C D E	38. A B C D E
9. A B C D E	19. A B C D E	29. A B C D E	39. A B C D E
10. A B C D E	20. A B C D E	30. A B C D E	40. A B C D E