Practice Exam #2

SECTION I

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

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

```
System.out.println(17 / 5 % 3 + 17 * 5 % 3);
```

(A) 1
(B) 4
(C) 9
(D) 42
(E) 42.5

2. Assume that a and b are properly declared and initialized boolean variables. Consider the following expression.

```
!(!a || b) || (!a && b)
```

When does the expression evaluate to true?

(A) When a and b have different values
(B) When a and b have the same value
(C) When both a and b have the value true and only then
(D) When both a and b have the value false and only then
(E) Never

3. Consider the following statement.

```
System.out.println("yes\\no".indexOf("\no"));
```

What is printed when the statement is compiled and executed?

(A) 3
(B) 4
(C) 5
(D) -1
(E) Nothing, due to a syntax error

4. Consider the following method.

```
public void printSomething(String s)
{
  int n = s.length();
  if (n < 1)
    return;
  String s1 = s.substring(1, n);
  printSomething(s1);
  System.out.println(s);
  printSomething(s1);
}
```

How many A's and how many letters total will be printed when `printSomething("ABCD")` is called?

	A's	Total
(A)	1	10
(B)	4	10
(C)	1	26
(D)	4	26
(E)	15	26

5. What are the smallest and the largest possible values of `x` after the following statement has been executed?

```
int x = (int)(Math.sqrt(4*Math.random()) + 0.5);
```

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

6. Which of the following expressions evaluate to `true`?

I. `"Boston".equals(new String("Boston"));`
II. `"Bos" + "ton" == new String("Boston");`
III. `(new String("Boston")).substring(3) == "ton";`

(A) I only
(B) I and II only
(C) II and III only
(D) I, II, and III
(E) None of the three

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

```
double pi = 3.14159;
int r = 100;
int area = (int)(pi * Math.pow(r, 2));
System.out.println(area);
```

(A) 30000
(B) 31415
(C) 31416
(D) 314159
(E) Depends on the particular computer system

8. Assume that the `String` variables `str1` and `str2` have been properly declared and initialized. Which of the following conditions correctly tests whether the value of `str1` is greater than or equal to the value of `str2` (in lexicographical order)?

(A) `str1 >= str2`
(B) `str1.compareTo(str2) >= 0`
(C) `str1.compareTo(str2) == true`
(D) `str1.length() > str2.length() || str1 >= str2`
(E) `str1.equals(str2) || str1.compareTo(str2) == 1`

9. Consider the following recursive method.

```
public static int fun(int n)
{
  int product = 2;
  for (int k = 2; k < n; k++)
    product *= fun(k);
  return product;
}
```

What does `fun(5)` return?

(A) 2
(B) 24
(C) 120
(D) 256
(E) Nothing. The method exceeds the maximum allowed recursion depth and the program is aborted, because the method does not specify a base case.

10. Consider the following class `Test` with an incomplete `main` method.

```
public class Test
{
  private static int year = 2020;

  public Test(int yr) { year = yr; }

  public void increment() { year++; }

  public String toString() { return year + ""; }

  public static void main(String[] args)
  {
    < missing statements >
  }
}
```

Which of the following replacements of < *missing statement*s > will result in a syntax error or print something other than `2020` when `main` is executed?

(A)
```
System.out.println(this);
```
(B)
```
Test x = new Test(2020);
System.out.println(x);
```
(C)
```
Test x = new Test(20);
System.out.println("" + x + x);
```
(D)
```
year = 2020;
System.out.println(year);
```
(E)
```
Test x = new Test(2019);
x.increment();
System.out.println(x);
```

11. Which of the following statements print `1234`?

I. `System.out.print(12 * 100 + 34);`
II. `System.out.print("12" + 34);`
III. `System.out.print(1 + "2" + 3 + 4);`

(A) None of the above
(B) I only
(C) I and II only
(D) II and III only
(E) I, II, and III

12. Consider the following code segment.

```
int i = 2;
for (int k = 0; k <= 12; k += i)
{
  System.out.print(k + " ");
  i++;
}
```

What is printed when the statement is executed?

(A) 0 3 7
(B) 0 3 7 12
(C) 2 5 8 11
(D) 0 3 6 9 12
(E) 0 2 4 6 8 10

13. Consider the following class.

```
public class Monster
{
  private String x;

  public void feed(String number)
  {
    if (x == null || x.compareTo(number) < 0)
      x = number;
  }

  public String get()
  {
    return x;
  }
}
```

What is printed when the following code in `Monster`'s client class is executed?

```
Monster bfg = new Monster();
String[] numbers = {"One", "Two", "Three", "Four", "Five",
                    "Six", "Seven", "Eight", "Nine", "Ten"};
for (String number : numbers)
  bfg.feed(number);
System.out.println(bfg.get());
```

(A) One
(B) Two
(C) Eight
(D) Ten
(E) NullPointerException

14. What is printed when the code segment is executed?

```
int[] a = {0, 1};
int[] b = a;
a[0] = 1;
b[0] = 2;
System.out.println(a[0] + b[0] + a[1] + b[1]);
```

(A) 3
(B) 4
(C) 5
(D) 6
(E) None of the above

15. What are the values of `a` and `b` after two iterations through the `for` loop?

```
int a = 1, b = 2;
for (int k = 1; k <= 20; k++)
{
  int oldA = a;
  a = b;
  b = (1 + b)/oldA;
}
```

(A) 2 and 2
(B) 2 and 3
(C) 3 and 2
(D) 3 and 3
(E) 3 and 6

16. Consider the following method.

```
public void mystery(int a, int b)
{
  System.out.print(a + " ");
  if (a <= b)
    mystery(a + 5, b - 1);
}
```

What is printed when `mystery(0, 16)` is called?

(A) 0
(B) 0 5
(C) 0 5 10
(D) 0 5 10 15
(E) 0 5 10 15 20

17. Consider the following method.

```
/** Precondition: a != null; a.length > 0 */
private static void doIt(double[] a)
{
  for (int k = 0; k < a.length / 2; k++)
  {
    double temp = a[k];
    a[k] = a[a.length - 1 - k];
    a[a.length - 1 - k] = temp;
  }
}
```

Which of the following best describes the task performed by this method?

(A) Sorts an array in ascending order
(B) Sorts an array in descending order
(C) Swaps the first and last elements of an array
(D) Reverses the order of elements in an array
(E) None of the above tasks is implemented correctly

18. Assume that the `int` variables `a` and `b` have been properly declared and initialized. For which of the following pairs of values of `a` and `b` is the value of the following expression `true`?

```
(a > 20 && a < b) || (a > 10 && a > b)
```

(A) 5 and 0
(B) 5 and 10
(C) 15 and 10
(D) 15 and 20
(E) None of the above

19. Brad has derived his class from the library class `JPanel`. `JPanel`'s `paintComponent` method prints a blank picture in a panel. Brad has overridden `JPanel`'s `paintComponent` to display his own picture. Brad's class compiles with no errors, but when he runs the program, only a blank background is displayed. Which of the following hypotheses CANNOT be true in this situation?

(A) Brad misspelled "paintComponent" in his method's name.
(B) Brad specified an incorrect return type for his `paintComponent` method.
(C) Brad chose the wrong type for a parameter in his `paintComponent` method.
(D) Brad specified two parameters for his `paintComponent` method, while `JPanel`'s `paintComponent` takes only one parameter.
(E) Brad has a logic error in his `paintComponent` code that prevents it from generating the picture.

20. Consider the following code segment. It is supposed to calculate and print the sum $1 + 2 + ... + 20$.

```
int count = 0, sum = 0;
while (count < 20)
{
  sum += count;
}
System.out.println(sum);
```

Which of the following statements best describes the result?

(A) The total printed will be correct.
(B) The total printed will be too small by 20.
(C) 0 is printed.
(D) 20 is printed.
(E) Nothing is printed, because the program goes into an infinite loop.

21. Consider the following code segment.

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

for (int i = 1; i <= 8; i++)
{
  list.add(i);
}

for (int i = 0; i < list.size(); i++)
{
  list.remove(i);
}

for (Integer x : list)
{
  System.out.print(x + " ");
}
```

What is printed when the code segment is executed?

(A) `IndexOutOfBoundsException`
(B) `1 3 5 7`
(C) `2 4 6 8`
(D) `1 2 3 4 5 6 7 8`
(E) No output, because the resulting list is empty

22. Consider the following code segment with a missing `for` loop.

```
ArrayList<String> letters = new ArrayList<String>();

letters.add("A");
letters.add("B");
letters.add("C");

< missing "for" loop >

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

The code segment, when executed, prints

```
[A*, B*, C*]
```

Which of the following can replace < *missing "for" loop* >?

I.
```
for (int i = 0; i < letters.size(); i++)
{
  letters.set(i, letters.get(i) + "*");
}
```

II.
```
for (int i = 0; i < letters.size(); i++)
{
  String s = letters.get(i);
  s = s + "*";
}
```

III.
```
for (String s : letters)
{
  s = s + "*";
}
```

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

Questions 23 and 24 refer to a project that includes the following classes.

```
public class Cake
{
  private String name;
  private int price;

  public Cake(String _name, int _price)
  {
    name = _name;
    price = _price;
  }

  public int getPrice() { return price; }

  < other constructors and methods not shown >
}

public class Sale
{
  private ArrayList<Cake> items;

  public Sale() { items = new ArrayList<Cake>(); }
  public void add(Cake cake) { items.add(cake); }

  public int getTotal()
  {
    int total = 0;

    for (Cake cake : items)
      total += cake.getPrice();

    return total;
  }
}
```

The project designer has instructed the programmer to modify the code as follows: to introduce

```
public class PricedItem
{
  int getPrice() { return 0; }
}
```

into the project, add `extends PricedItem` to the header of the class `Cake`, and replace `Cake` with `PricedItem` everywhere in the `Sale` class.

23. Which design principle is applied here, and which Java feature makes it possible for the modified code to work?

(A) Modularity and recursion
(B) Modularity and encapsulation
(C) Abstraction and encapsulation
(D) Encapsulation and polymorphism
(E) Abstraction and polymorphism

24. Which of the following are good reasons for this change?

I. In a future version of the project, the `items` list in a `Sale` object may hold items of the type of a subclass of `Cake`.

II. In a future version of the project, different types of `PricedItem` objects can be intermixed in the `items` list in a `Sale` object.

III. The `Cake` class can be reused in other projects dealing with a different type of `PricedItem` items.

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

25. Consider the following code segment.

```
ArrayList<String> list = new ArrayList<String>();
list.add("A");
list.add("B");
list.add("C");
for (String s : list)
{
  String t = list.get(list.size() - 1);
  list.set(list.size() - 1, s);
  s = t;
}
```

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

(A) `["A", "B", "C"]`
(B) `["C", "B", "A"]`
(C) `["C", "A", "B"]`
(D) `["C", "C", "C"]`
(E) `["A", "B", "B"]`

26. Consider the following class.

```
public class FrequentFlyer
{
  private int miles = 0;

  public FrequentFlyer(int m) { miles = m; }
  public void addMiles(int m) { miles += m; }
  public int getMiles() { return miles; }
}
```

What is printed when the following code segment in a client class is executed?

```
FrequentFlyer akshay = new FrequentFlyer(20000);
FrequentFlyer bruno = new FrequentFlyer(10000);
FrequentFlyer cindy = new FrequentFlyer(0);
FrequentFlyer[] friends = {akshay, bruno, cindy};
int total = 0;
for (FrequentFlyer p : friends)
{
  p.addMiles(1000);
  total += p.getMiles();
}
System.out.println(total);
```

(A) 0
(B) 3000
(C) 30000
(D) 33000
(E) None of the above

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

```
String url = "http://www.usa.gov";
int pos = url.indexOf("http://");
if (pos >= 0)
{
  System.out.println("<" + url.substring(0, pos) + ">");
}
else
{
  System.out.println("not found");
}
```

(A) <www.usa.gov>
(B) <http://www.usa.gov>
(C) <>
(D) not found
(E) StringIndexOutOfBoundsException

28. The statement

```
Animal a = new Mammal("Elephant");
```

compiles with no errors. Which of the following situations will permit that?

(A) `Mammal` is a class with a constructor that takes one parameter of the `String` type, and `Animal` is its subclass.
(B) `Animal` is a class with a constructor that takes one parameter of the `String` type, and `Mammal` is its subclass that has no constructors defined.
(C) `Mammal` is a class with a constructor that takes one parameter of the `String` type, and `Animal` is a superclass of `Mammal`.
(D) `Animal` has a public static data field `String Mammal`.
(E) None of the above

29. Consider the following code segment.

```
ArrayList<String> list = new ArrayList<String>();
list.add("One");
list.add("Two");
String[] msg = new String[2];
list.add(msg[0]);
< another statement >
```

Which of the following choices for < *another statement* > will cause a `NullPointerException` when the code is compiled and executed?

(A) `msg[0] = "Three";`
(B) `msg[0] = list.get(list.size());`
(C) `msg[1] = msg[0].substring(0, 2);`
(D) `list.add(2, msg[0]);`
(E) `if (!"Three".equals(list.get(2))) msg[0] = "Three";`

30. At a county fair, prizes are awarded to the five heaviest cows. More than 2000 cows are entered, and their records are stored in an array. Which of the following algorithms provides the most efficient way of finding the records of the five heaviest cows?

(A) Selection Sort
(B) Insertion Sort
(C) Mergesort
(D) Insertion Sort terminated after the first five iterations
(E) Selection Sort terminated after the first five iterations

Questions 31 and 32 refer to the following class.

```
public class Sample
{
  private double[][] amps;

  public Sample(int n)
  {
    < missing statements >
  }

  public double get(int j, int k)
  {
    return amps[j][k];
  }
}
```

31. Which of the following code segments can replace < *missing statements* > in `Sample`'s constructor so that it initializes `amps` to hold a table of values with `n` rows and `n` columns and fills them with random values $0.0 \leq$ `amps[j][k]` < 1.0?

I.
```
amps = new double[n][n];
```

II.
```
amps = new double[n][n];
for (int j = 0; j < n; j++)
{
  for (int k = j; k < n; k++)
  {
    amps[j][k] = Math.random();
    amps[k][j] = Math.random();
  }
}
```

III.
```
amps = new double[n][n];
for (double[] r : amps)
{
  for (int k = 0; k < n; k++)
  {
    r[k] = Math.random();
  }
}
```

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

32. Given

```
int size = 100;
Sample s = new Sample(size);
```

which of the following statements in a client class of `Sample` assigns to `x` the value in the last row and the first column of `amps` in `s`?

(A) `double x = s[99][0];`
(B) `double x = s.get(size - 1, 0);`
(C) `double x = s.get[s.length - 1, 0];`
(D) `double x = s.get(s.amps.length - 1, 0);`
(E) `double x = s.amps[s.amps.length - 1][0];`

33. Consider an incomplete definition of the class `C`.

```
public class C
{
  private int value;

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

Suppose we have a method

```
public static int compare(C x, C y)
{
  return x.value - y.value;
}
```

and we need to find a "home" for it: place it into some class. Where can we place this method so that it compiles with no errors?

(A) Only into `C`
(B) Only into `C` or any subclass of `C`
(C) Only into `C` or any superclass of `C`
(D) Into any class
(E) This method will always cause a syntax error, no matter what class we place it in.

34. Consider the following class.

```
public class ArrayProcessor
{
  public static void run(int[] arr)
  {
    for (int i = 0; i < arr.length; i++)
    {
      for (int j = arr.length - 1; j > i; j--)
      {
        if (arr[j] < arr[i])
        {
          swap(arr, i, j);
        }
      }
    }
  }

  private static void swap(int[] arr, int i, int j)
  {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
  }
}
```

How many times will `ArrayProcessor`'s `swap` method be called when the following code segment is executed?

```
int[] counts = {1, 2, 3, 4, 5, 0};
ArrayProcessor.run(counts);
```

(A) 5
(B) 10
(C) 15
(D) 30
(E) 35

35. Suppose it takes about 18 milliseconds to sort an array of 80,000 random numbers using Mergesort. Suppose for an array of 160,000 numbers, Mergesort runs for 40 milliseconds. For approximately how much time will Mergesort run on an array of 320,000 numbers? Choose the closest estimate.

(A) 88 milliseconds
(B) 96 milliseconds
(C) 126 milliseconds
(D) 160 milliseconds
(E) 192 milliseconds

36. The Binary Search algorithm is designed to work with an array sorted in ascending order. Under which of the following circumstances will the algorithm find a given target value even if the array is not sorted?

    I. The array has an odd number of elements and the target value is located exactly in the middle of the array.

    II. The array is partially sorted: the left third of the array has values all in ascending order and the target value is among them.

    III. The array is partially sorted: all the values to the left of the target are smaller than the target, and all the values to the right of the target are larger than the target.

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

37. Consider the following code segment.

```
ArrayList<Integer> lst = new ArrayList<Integer>();
int k = 2;

while (lst.size() < 5)
{
  boolean found = false;

  for (Integer n : lst)
    if (k % n == 0)
      found = true;

  if (!found)
    lst.add(k);

  k++;
}

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

What is printed when the code segment is executed?

(A) `[2, 3, 4, 5, 6]`
(B) `[2, 3, 5, 7, 11]`
(C) `[2, 3, 4, 5, 6, 7]`
(D) `[2, 3, 5, 7, 11, 13]`
(E) Nothing is printed — the program goes into an infinite loop.

Questions 38 and 39 refer to the following class `Game` and the incomplete class `ChessGame`.

```
public class Game
{
  private String gameName;
  private ArrayList<String> players;

  public Game(String name)
  {
    gameName = name;
    players = new ArrayList<String>();
  }

  public Game(String name, String[] people)
  {
    gameName = name;
    players = new ArrayList<String>();
    for (String nm : people)
      players.add(nm);
  }

  public void addPlayer(String name)
  {
    players.add(name);
  }

  public String getPlayer(int k)
  {
    return players.get(k - 1);
  }

  public String toString()
  {
    return gameName + " game " + players.toString();
  }
}


public class ChessGame extends Game
{
  public ChessGame(String black, String white)
  {
    < missing code >
  }
}
```

38. Consider the following code segment in a client class of `Game`.

```
String[] players = {"Annette", "Bertrand",
                    "Claude", "Danielle"};

Game game = new Game("Bauernschnapsen", players);

System.out.println( < missing expression >);
```

Which of the following can replace < *missing expression* > so that the code prints `"Annette"`?

(A) `game.getPlayer(0)`
(B) `game.getPlayer(1)`
(C) `game.players.get(0)`
(D) `game.players.get(1)`
(E) `game.getPlayers().get(0)`

39. Suppose the statement

```
System.out.println(new ChessGame("Deep Blue", "Kasparov"));
```

prints

```
Chess game [Deep Blue, Kasparov]
```

Which of the following can replace < *missing code* > in `ChessGame`'s constructor?

I.
```
super("Chess", black, white);
```

II.
```
super("Chess");
super.addPlayer(black);
super.addPlayer(white);
```

III.
```
String[] players = {black, white};
super("Chess", players);
```

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

40. Consider the following three implementations of a method `rotate90degrees` and the code segments (in the same class) used to test them.

I.
```
public int[] rotate90degrees(int[] v)
{
  int[] w = new int[2];
  w[0] = -v[1];
  w[1] = v[0];
  return w;
}

  int[] v = {1, 2};
  v = rotate90degrees(v);
  System.out.println(v[0] + ", " + v[1]);
```

II.
```
public void rotate90degrees(int[] v)
{
  int  temp  = v[0];
  v[0] = -v[1];
  v[1] = temp;
}

  int[] v = {1, 2};
  rotate90degrees(v);
  System.out.println(v[0] + ", " + v[1]);
```

III.
```
public int[] rotate90degrees(int[] v)
{
  int  temp  = v[0];
  v[0] = -v[1];
  v[1] = temp;
  return v;
}

  int[] v = {1, 2};
  rotate90degrees(v);
  System.out.println(v[0] + ", " + v[1]);
```

Which of these implementations will compile with no errors and print `-2, 1`?

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

Practice Exam #2

SECTION II

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

1. Many websites have strict requirements for login passwords. The class `Password` provides methods that help verify that the password requirements are met and can also generate a password from a secret phrase and a number.

```
public class Password
{
  /** Returns true if password contains an uppercase letter;
   *  otherwise returns false.
   */
  private static boolean hasUpper(String password)
  { /* implementation not shown */ }

  /** Returns true if password contains a lowercase letter;
   *  otherwise returns false.
   */
  private static boolean hasLower(String password)
  { /* implementation not shown */ }

  /** Returns true if password contains a digit;
   *  otherwise returns false.
   */
  private static boolean hasDigit(String password)
  { /* implementation not shown */ }

  /** Returns true if password is valid, as described
   *  in part (a)
   */
  public static boolean isValid(String  password)
  {
    /* to be implemented in part (a) */
  }

  /** Returns a string made of the first letters of the words
   *  in phrase.
   *  Precondition: phrase is not null.
   */
  public static String makePasswordLetters(String phrase)
  { /* implementation not shown */ }
```

Continued

```
    /** Generates a password from phrase and n, as described in
     *  part (b).
     */
    public static String makePassword(String phrase, int n)
    { /* to be implemented in part (b) */ }

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

(a) Write a method `isValid` that checks whether the given string meets the following password requirements: has between 8 and 16 characters; contains an uppercase letter; contains a lowercase letter; contains a digit. For example, `"Tttoajaf0"` is a valid password; `"TrdiaywAsIcntb5191"` is not valid, because it is too long — more than 16 characters. Use the methods provided in the `Password` class to determine whether a given string includes an uppercase letter, a lowercase letter, and a digit; you will not receive full credit if you write your own code to check for these occurrences.

Complete the method `isValid` below.

```
/** Returns true if password is 8 to 16 characters long and
 *  includes an uppercase letter, a lowercase letter, and a
 *  digit; otherwise returns false.
 *  Precondition: password is not null.
 */
public static boolean isValid(String password)
```

(b) Many internet users find it difficult to remember passwords for various websites. So they take a secret phrase and make a string of the first letters of the words in that string. `Password`'s method `makePasswordLetters(phrase)` (whose implementation is not shown) returns the string of the first letters of the words in a given phrase. For example,

```
makePasswordLetters("Then took the other, as just as fair")
```

returns `"Tttoajaf"`.

The method `makePassword(phrase, n)` takes a string of the first letters of the words in `phrase` and appends to it the digits of `n` taken in reverse order. So if `phrase` is `"Then took the other, as just as fair"`, then `makePassword(phrase, 1915)` returns `"Tttoajaf5191"`. `makePassword` returns `null` if the password generated that way is not a valid password.

Complete the method `makePassword` below.

```
/** Generates a password by taking the first letters
 *  of the words in phrase followed by the digits of n
 *  taken in reverse order. Returns the generated
 *  password if it is valid; otherwise returns null.
 *  Precondition: phrase is a non-empty string; n >= 0.
 */
public static String makePassword(String phrase, int n)
```

For additional practice, write the `makePasswordLetters` method. The consecutive words in `phrase` are separated by one or several spaces. Also add the necessary `String` constants for uppercase and lowercase letters and for digits and implement the `hasUpper`, `hasLower`, and `hasDigit` methods. Once your `Password` class is complete, test your code written in Parts (a) and (b).

2. The class `Shopping` helps a consumer choose the best option for buying a particular item. `Shopping`'s constructor initializes an `int` constant that represents the maximum distance (in miles) the consumer is willing to travel to a store. The `Shopping` class has three methods:

   - `addShoppingOption` — presents a shopping option to the consumer.
   - `streetPrice` — returns the average price of the item (a `double`) across all the shopping options presented to the consumer (regardless of the distance to the store), rounded to the nearest cent.
   - `toString` — returns a string that combines the store name, the item price at that store, and the distance from the consumer's home for the best shopping option (that is, the store within the maximum distance that offers the best price)

   `addShoppingOption` takes three parameters: the store name (a `String`), the item price (a `double`), and the distance from the consumer's home to that store (an `int`).

   For example, the following statements in a client of the `Shopping` class —

```
Shopping kayak = new Shopping(20);
kayak.addShoppingOption("JMart", 225.95, 18);
kayak.addShoppingOption("Jest Buy", 189.00, 24);
kayak.addShoppingOption("Tarjet", 220.95, 19);
System.out.println(kayak);
System.out.println(kayak.streetPrice());
```

   — print

```
Tarjet 220.95 19
211.97
```

   because the best shopping option presented to the consumer, within the 20-mile radius, is Tarjet, at $220.95, and the average price for the three shopping options, $(225.95 + 189.00 + 220.95) / 3$, rounded to the nearest cent, is $211.97.

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

   For additional practice, allow the consumer to order the item online, using 0 distance in `addShoppingOption` calls. For "brick-and-mortar" (not online) shopping options, add to the item price the cost of gas at $0.45 per mile for traveling to the store and back.

   Another possible project is to allow the consumer to shop for three items at once, replacing the price parameter in the `addShoppingOption` method and the return value of the `streetPrice` method with an array of three `double` values and recommending the best option based on the sum of the prices for the three items.

3. Memory Lane is a one-dimensional version of the popular card matching game Memory. Several cards are arranged face down in a row on the table. Each card contains a word or a picture, and there are two identical cards for each word or picture (so the number of cards is even). A player opens any two cards; if they have the same word or picture on them, the player removes both cards and shifts the remaining cards to the left to fill the gaps. If the cards are different, the player puts them back, face down.

   In the computer model of the game, a card is represented by an object of the class `Card`. The `Card` class has a constructor that takes a `String` as a parameter and stores the string in an instance variable. The `Card` class also defines an `equals` method, which compares this card to another based on the strings they hold.

   The class `MemoryLane` supplies several methods for playing the game. A partial implementation of this class is shown below.

```
public class MemoryLane
{
  /** The array of cards */
  private Card[] cards;

  /** The number of cards remaining to pair up */
  private int numCards;

  /** Returns true if for each card among the first
   *  numCards cards in the cards array there is exactly one
   *  other card equal to the first one; otherwise
   *  returns false.
   */
  public boolean isValidArrangement()
  { /* to be implemented in part (a) */ }

  /** Removes the card with index k from the cards array,
   *  shifting the subsequent cards "to the left" (toward the
   *  the beginning of the cards array) to close the gap;
   *  reduces numCards by one.
   *  Precondition: 0 <= k < numCards
   */
  public void removeCard(int k)
  { /* to be implemented in part (b) */ }

  /** If cards with the indices k1 and k2 are equal, removes
   *  these two cards and returns true; otherwise leaves
   *  the cards array unchanged and returns false.
   *  Precondition: 0 <= k1 < k2 < numCards
   */
  public boolean openTwoCards(int k1, int k2)
  { /* to be implemented in part (c) */ }

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

(a) Write a `boolean` method `isValidArrangement` that checks whether `numCards` cards at the beginning of the array `cards` represent a valid configuration for the game, that is, for each card there is exactly one other card equal to it.

Complete the method `isValidArrangement` below.

```
/** Returns true if for each card among the first
 *  numCards cards in the cards array there is exactly one
 *  other card equal to the first one; otherwise
 *  returns false.
 */
public boolean isValidArrangement()
```

(b) Write a method `removeCard` that removes a card with the given index from the array `cards`, shifting the subsequent cards, if any, by one position toward the beginning. The method also decrements `numCards` by 1.

Complete the method `removeCard` below.

```
/** Removes the card with index k from the cards array,
 *  shifting the subsequent cards by one toward the
 *  beginning to close the gap; reduces numCards by one.
 *  Precondition: 0 <= k < numCards
 */
public void removeCard(int k)
```

(c) Write a method `openTwoCards` that "opens" two cards with given indices. If the cards are equal, the method removes both cards and returns `true`; otherwise the array of cards remains unchanged and the method returns `false`. In writing this method, assume that the method `removeCard` from Part (b) works as specified, regardless of what you wrote there. You may not receive full credit if you duplicate `removeCard`'s code here.

Complete the method `openTwoCards` below.

```
/** If the cards with the indices k1 and k2 are equal,
 *  removes these two cards and returns true; otherwise
 *  leaves the cards array unchanged and returns false.
 *  Precondition: 0 <= k1 < k2 < numCards
 */
public boolean openTwoCards(int k1, int k2)
```

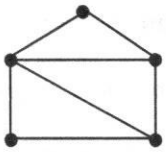
For additional practice, write the `Card` class. In addition to the constructor and the `equals` method explained in the introduction to this question, supply a `toString` method that returns the word on the card.

Also write and test a constructor for the `MemoryLane` class:

```
public MemoryLane(ArrayList<String> words)
```

The constructor should initialize the array `cards` by producing a valid configuration of randomly ordered pairs of cards. The words on the cards come from the list `words`; two cards are created for each word in the list `words`, so the size of the array `cards` will be `2*words.size()`. `numCards` should be set to the same value. Write a helper method `shuffle` that arranges the elements of an array of cards in random order in such a way that each card can end up in any location with equal probability.

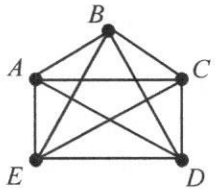
4. The picture below shows a few points connected by lines.



Such a structure is called a *graph*; the points are called *vertices*, and the lines are called *edges*. Graphs are useful for modeling all kinds of things: computer networks, airline routes, connections in social media apps, and board game rules and strategies, to name a few. We will consider only simple graphs in which any two vertices can be connected by at most one edge and a vertex cannot be connected to itself.

In some algorithms dealing with graphs, it is convenient to represent a graph as an n-by-n two-dimensional array of integers, where n is the number of vertices. Such a square array is called the *adjacency matrix* of the graph. If the vertices with indices `i` and `j` are connected by an edge, then `m[i][j]` and `m[j][i]` are both set to 1; otherwise these elements of `m` are set to 0 (where `m` is the name of the adjacency matrix).

(a) A simple graph is called *complete* if any two of its vertices are connected by an edge. For example:

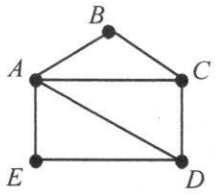


Write a `boolean` method `isComplete` that takes an adjacency matrix of a graph as a parameter and returns `true` if the corresponding graph is complete; otherwise the method returns `false`. (The adjacency matrix of a complete graph has 0's on the NW-SE diagonal and 1's everywhere else.)

Write the method `isComplete` below.

```
/** Returns true if the graph represented by the
 *  given adjacency matrix is a complete graph; otherwise
 *  returns false.
 */
public static boolean isComplete(int[][] m)
```

(b) To represent a graph in a computer program, we can label its vertices with numbers or letters. Here we will use the letters "A", "B", ... as labels and represent the edges as pairs of letters in one long string. For example:



Vertices: `["A", "B", "C", "D", "E"]`

Edges: `"AB BC DC DE AC AD EA"`

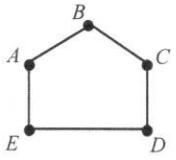
To find out whether an edge connects two vertices, we combine their letters into a two-letter string in two ways (first+second and second+first) and check whether one of these two combinations can be found in the string that represents the edges.

Write a method `makeAdjacencyMatrix` that takes two parameters, an `ArrayList` of vertices (single-letter strings) and a `String` of edges (two-letter substrings, separated by spaces), and returns the adjacency matrix of that graph.

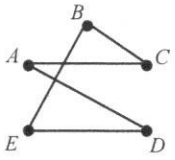
Complete the method `makeAdjacencyMatrix` below.

```
/** Returns the adjacency matrix of the graph represented by
 *  a list of its vertices and a string of its edges. The
 *  vertices are single letters, and the edges are
 *  two-letter substrings of edges, separated by spaces.
 */
public static int[][] makeAdjacencyMatrix(
                ArrayList<String> vertices, String edges)
```

A graph is called a *cycle* if its vertices are arranged in one cycle with each vertex connected to two neighbors. For example:



or



For additional practice, write a `boolean` method `isCycle` that takes the adjacency matrix of a graph and determines whether the graph is a cycle.