Questions 14-16 refer to the Point, Quadrilateral, and Rectangle classes below:

```
public class Point {
    private int xCoord;
    private int yCoord;
    // constructor
    public Point(int x, int y) {
        ….
    }
    // accessors
    public int get_x() {
        ….
    }
    public int get_y() {
        ….
    }
    // other methods not shown…
}
public abstract class Quadrilateral {
     private String myLabels;  // e.g., “ABCD”

     public Quadrilateral(String labels) {     myLabels = labels;     }
     public String getLabels() {     return myLabels;    }
     public abstract int perimeter();
     public abstract int area();
}
```

```
public class Rectangle extends Quadrilateral {
    private Point myTopLeft;  // coords of top left corner
    private Point myBotRight;  // coords of bottom right corner

    //constructor
    public Rectangle (String labels, Point topLeft, Point botRight) {
        /*  implementation code */
    }

    public int perimeter() {
        /*  implementation code */
    }

    public int area() {
        /*    implementation code */
    }
    // other methods not shown…
}
```

14. Which statement about the Quadrilateral class is false?
(A) The perimeter and area methods are abstract because there's no suitable default code for them.
(B) The getLabels method is not abstract because any subclasses of Quadrilateral will have the same code for this method.
(C) If the Quadrilateral class is used in a program, it must be used as a superclass for at least one other class.
(D) No instances of a Quadrilateral object can be created in a program.
(E) Any subclasses of the Quadrilateral class must provide implementation code for the perimeter and area methods.

15. Which represents correct /* implementation code */ for the Rectangle constructor?

```
I  super(labels);

II  super(labels, topLeft, botRight);

III  super(labels);
     myTopLeft = topLeft;
     myBotRight = botRight;
```

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

16. Refer to the Parallelogram and Square classes below.

```
public class Parallelogram extends Quadrilateral {
    // private instance variables and constructor not shown…
   public int perimeter() { /* implementation code */ }
   public int area() { /* implementation code */ }
}
public class Square extends Rectangle {
   // private instance variables and constructor not shown
   public int perimeter() { /* implementation code */ }
   public int area() { /* implementation code */ }
}
```

Consider an ArrayList<Quadrilateral> quadList whose elements are of type Rectangle, Parallelogram, or Square. Refer to the following method, writeAreas:

```
/* Precondition: quadList contains Rectangle, Parallelogram, or Square objects in an unspecified order
*/
public static void writeAreas(ArrayList quadList) {
    for(Quadrilateral quad : quadList)
        System.out.println(“Area of ” + quad.getLabels() + “ is ” + quad.area());
}
```

What is the effect of executing this method?

(A) The area of each Quadrilateral in quadList will be printed.

(B) A compile-time error will occur, stating that there is no area method in abstract class Quadrilateral.

(C) A compile-time error will occur, stating that there is no getLabels method in classes Rectangle, Parallelogram, or Square.

(D) A NullPointerException will be thrown.

(E) A ClassCastException will be thrown.