635006

```
1.  public BallPanel extends javax.swing.JPanel {
2.     private Ball[] _balls;
3.     public BallPanel(int numberOfBalls){
4.        ??????
5.        for (int i=0;i<_balls.length;i++)
6.           _balls[i] = new Ball();
7.     }
8.  }
```

We need to add something at line 4 above to avoid an error in line 5. Which of the following line 4 expressions fix the error in line 5?

a. `super();`
b. `_balls.setLength(numberOfBalls);`
*c. `_balls = new Ball[numberOfBalls];`
d. `_balls.length = numberOfBalls;`

e. B, C, and D all work.

f. "
g. "
h. "
i. "
j. "

General Feedback:

The problem with line 5 (without adding line 4) is that _balls is declared, but not instantiated as an array, so _balls is null. The code compiles fine (assuming you have a class Ball that has a constructor with no parameters), but will crash at run time if you instantiate a BallPanel.

B and D wouldn't help: B would crash at run time for the same reason as 5, and D gives a compile-time error since length is not directly settable for a Java array. A would work, but would have no effect on the problem.

633279
Which of the following makes an appropriate pre-condition for this code?

```
double calculate_percentage(double sum, double n)
{
    // calculates and returns a percentage given a sum and a length (n)
    // <missing pre-condition>
    // POST: none
    return (sum / n) * 100;
}
```

a. PRE: sum < 100

*b. PRE: n != 0

c. PRE: sum != NULL and n != NULL

d. PRE: none

e.
f. "
g. "
h. "
i. "

General Feedback:

The code will faill if n = 0, as it will divide by 0.

629623
Consider writing a program to be used by a farm to keep track of information about the fruits and vegetables they sell. For each sale, they would like to keep track of the type of item (eggplant, tomato, etc.), the quantity sold, the price, and the market at which the sale was made. Which of the following is the best way to represent the information?

*a. Define one class, FarmSale, with four fields: type, quantity, price, and market.

b. Define one superclass, FarmSale, with four subclasses: Type, Quantity, Price, and Market.

c. Define five unrelated classes: FarmSale, Type, Quantity, Price, and Market.

d. Define five classes: FarmSale, Type, Quantity, Price, and Market.. Make Market a subclass of Price, make Price a subclass of Quantity, make Quantity a subclass of Type, and make Type a subclass of FarmSale.

e. Define five classes: FarmSale, Type, Quantity, Price, and Market. Make FarmSale a subclass of Type, make Type a subclass of Quantity, make Quantity a subclass of Price, and make Price a subclass of Market.

f. "
g. "
h. "
i. "
j. "

General Feedback:

Which classes you define depends generally on the needs of the software. It might be good to have `Market` be a class of its own, with the address of the market, contact information for the organizers, etc. The item type might be modeled with a `Product` class, with information about the date it was planted, for example.

But of the choices given, A is the best. We can rule out choices B, C, D, and E because price and quantity don't need data and associated methods of their own and are most appropriately modeled as fields in the `FarmSale` class. That leaves A.