

My Courses All Courses About Contact



Help vitorpbarbosa7



Course Progress

Course > Readings/Videos > Reading 10: Abstraction Functions and Rep Invariants > Questions



Questions

□ Bookmark this page

rep exposure

1/1 point (graded)

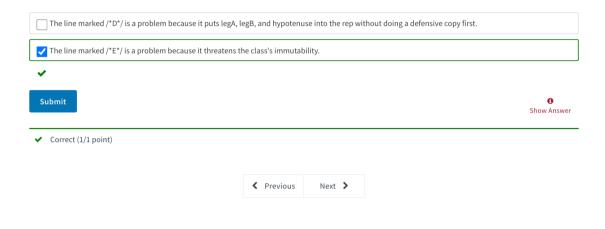
Consider the following problematic datatype:

```
/** Represents an immutable right triangle. */
     class RightTriangle {
        private double[] sides;
          // sides[0] and sides[1] are the two legs,
          // and sides[2] is the hypotenuse, so declare it to avoid having a
          // magic number in the code:
/*B*/
          public static final int HYPOTENUSE = 2;
          /** Make a right triangle.
           * @param legA, legB \, the two legs of the triangle
          * @param hypotenuse the hypotenuse of the triangle.

* Requires hypotenuse^2 = legA^2 + legB^2
*C*
                       (within the error tolerance of double arithmetic)
          public RightTriangle(double legA, double legB, double hypotenuse) {
/*D*/
              this.sides = new double[] { legA, legB, hypotenuse };
          /\!*\!* Get all the sides of the triangle.
          * @return three-element array with the triangle's side lengths
          public double[] getAllSides() {
/*E*/
              return sides;
          /** @return length of the triangle's hypotenuse */
          public double getHypotenuse() {
             return sides[HYPOTENUSE];
          /** @param factor to multiply the sides by
     * @return a triangle made from this triangle by
          * multiplies all side lengths by factor.
          public RightTriangle scale(double factor) {
             return new RightTriangle (sides[0]*factor, sides[1]*factor, sides[2]*factor);
          /** @return a regular triangle made from this triangle.
          * A regular right triangle is one in which
          * both legs have the same length.
          public RightTriangle regularize() {
              double bigLeg = Math.max(side[0], side[1]);
              return new RightTriangle (bigLeg, bigLeg, side[2]);
     }
```

Which of the following statements are true? Check all that apply.

The line marked /*A*/ is a problem for rep exposure because arrays are mutable.	
✓ The line marked /*B*/ is a problem for representation independence because it reveals how the sides array is organized.	
The line marked *C* is a problem because creator operations should not have preconditions.	



△ ○ Some Rights Reserved

Open Learning Library Connect About Contact Accessibility Twitter All Courses Facebook Why Support MIT Open Learning? Help