CS230 assignment 2 UPI: qzhu496
All source file can be found here:
https://github.com/1and1get2/cs230_ass2

cs230_ass2 / Asst2 / Test / qwta / **QuizTest.java**

```java
package qwta;

import static org.junit.Assert.*;

import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import java.util.Vector;

public class QuizTest {

        private Quiz quiz1, quizMax, quizAvg, quizLast;
 private Vector<Integer> vector1 = new Vector<Integer>();
        private Course course = new Course("course_name", new Instructor(
                        "Instructor of a course"));
     private Scoresheet scoresheet;

        @Before
public void setUp() {
          quizMax = new Quiz("quizMax", course, 5, SummaryStatistic.MAX_SCORE);
          quizAvg = new Quiz("quizAvg", course, 5, SummaryStatistic.AVG_SCORE);
          quizLast = new Quiz("quizLast", course, 5, SummaryStatistic.LAST_SCORE);
               vector1.add(10);
               vector1.add(20);
               vector1.add(30);
               vector1.add(20);
               scoresheet = new Scoresheet(quizAvg, new Marksheet(new Student("student1"), course));
 }

      // coverage of the three methods for computing a SummaryStatistic on a
 // vector of integer scores
   @Test
 public void testSummariseScoresMax() {
        assertEquals((Integer) 30, quizMax.summariseScores(vector1));
 }

      @Test
 public void testSummariseScoresAvg() {
        assertEquals((Integer) 20, quizAvg.summariseScores(vector1));
 }

      @Test
 public void testSummariseScoresLast() {
       assertEquals((Integer) 20, quizLast.summariseScores(vector1));
 }

      // coverage of different lengths of the score vector
  @Test
 public void testSummariseScoresMoreElement() {
        // create a lot of element and add into vector1
       // UNVERIFIED_BUG if NUMBERS_OF_ELEMENT becomes large enough
          int NUMBERS_OF_ELEMENT = 1000000;
           Vector vector2 = new Vector<Integer>();
     int sum = 0, temp = 0;
      for (int i = 0; i < NUMBERS_OF_ELEMENT; i++) {
              temp =  (int)(1000000 * Math.random());
            sum += temp;
               vector2.add(temp);
          }
           assertEquals((Integer) (sum/NUMBERS_OF_ELEMENT), quizAvg.summariseScores(vector2));
  }

    // 1 c,d (none)integer values within the score vector
@Test
public void testSummariseScoresMoreMinusElement() {
         quiz1 = new Quiz("quiz1", course, 5, SummaryStatistic.AVG_SCORE);
           vector1.add((-30));
         // cause a error: JAVA_TYPE_SAFETY_SUCCESSFUL_TEST (minus integer)
          assertEquals((Integer) 20, quiz1.summariseScores(vector1));
  }

    // none-interger
    @Test
public void testSummariseScoresMoreNoneIntegerElement() {
           quiz1 = new Quiz("quiz1", course, 5, SummaryStatistic.AVG_SCORE);
           Vector vector2 = new Vector(vector1);
       vector2.add("hello");
       // cause a error: JAVA_TYPE_SAFETY_SUCCESSFUL_TEST UNVERIFIED_BUG
           assertEquals((Integer) 20, quiz1.summariseScores(vector2));
  }
```

```
      // none-interger, float number
@Test
 public void testSummariseScoresMoreNoneIntegerElementFloat() {
        quiz1 = new Quiz("quiz1", course, 5, SummaryStatistic.AVG_SCORE);
             Vector vector2 = new Vector();
        vector2.add(30);
              vector2.add(2.3);
             // cause a error: JAVA_TYPE_SAFETY_SUCCESSFUL_TEST UNVERIFIED_BUG
             assertEquals((Integer) 30, quiz1.summariseScores(vector2));
   }

      // none-interger, float number
@Test
 public void testSummariseScoresNull() {
       quiz1 = new Quiz("quiz1", course, 5, SummaryStatistic.AVG_SCORE);
             // Vector vector2 = new Vector();
             // vector2.add(30);
           vector1.add(null);
            // cause a error: JAVA_TYPE_SAFETY_SUCCESSFUL_TEST UNVERIFIED_BUG
             assertEquals((Integer) 20, quiz1.summariseScores(vector1));
   }

      // @Test
      // public void testSummariseScoresAvg() {
     // //assertEquals("testSummariseScores 1: ", 100,
     // quiz1.summariseScores(vector1));
   // assertEquals((Integer)20, quiz1.summariseScores(vector1));
 // }
```

```
        // test 3 attempts
      @Test
   public void testSitQuizAttemptThreeTimes(){
//           Scoresheet scoresheet= new Scoresheet(quizAvg, new Marksheet(new Student("student1"), course));
         assertEquals(0, quizAvg.sitQuiz(scoresheet));
           assertEquals(0, quizAvg.sitQuiz(scoresheet));
           assertEquals(0, quizAvg.sitQuiz(scoresheet));
           scoresheet.reportOnScoresForStudent();
          scoresheet.reportScoresToInstructor();
  }

        // test one more attempt
       @Test
   public void testSitQuizAttemptBeyondLimitedTimes(){
//           Scoresheet scoresheet= new Scoresheet(quizAvg, new Marksheet(new Student("student1"), course));
         assertEquals(0, quizAvg.sitQuiz(scoresheet));
           assertEquals(0, quizAvg.sitQuiz(scoresheet));
           assertEquals(0, quizAvg.sitQuiz(scoresheet));
           assertEquals(0, quizAvg.sitQuiz(scoresheet));
           assertEquals(0, quizAvg.sitQuiz(scoresheet));
           assertEquals(0, quizAvg.sitQuiz(scoresheet));
           scoresheet.reportOnScoresForStudent();
          scoresheet.reportScoresToInstructor();
  }

}
```

cs230_ass2 / Asst2 / qwta / **Quiz.java**

```
package qwta;
// BUGFIX
import java.util.Iterator;
import java.util.Vector;
import java.util.Collections;

public class Quiz {

        public String name;
     public Course assigned;
 public int maxAttempts; // limits the # of attempts per student
 public SummaryStatistic summaryMethod; // defines how marks are computed

        public Vector<Scoresheet> scored;

        public int maxScore = 10;               //highest value that QuestionMark will award to any student taking this
quiz.
   public double maxMarks = 0.6;   //scale a student's summarised score
//  scaledMarks() = summarisedScore / maxScore * maxMarks.

        public int sitQuiz(Scoresheet ss) {
             // In an included QM use-case, our student would get a new score
                // whenever they sit ss.scored,
         // but our simulated students always get a zero score from our simulated
                // QM!
          int score = 0;
          // QuizWhiz now adds the new score to the student's scoresheet
          ss.addScore(score);
             return score;
   }
```

```
        // BUGFIX change return type from Integer to float
      public Integer summariseScores(Vector<Integer> sv) {
            //  detect all none integer element and delete them
             String exceptionMsg = "";
               int i = 0;
              try {
                   for (i = 0; i < sv.size(); i++) {
//                              BUGFIX detect null element and remove it from the vector
                               if (sv.get(i) == null) {
                                      sv.remove(i);
                                  exceptionMsg = "null element at index: " + i;
                                  throw new NullPointerException(exceptionMsg);
                              }
                               //  BUGFIX detect negative number and remove it from the vector
                               if (sv.get(i) < 0) {
                                   int nagetiveNum = sv.remove(i);
                                 exceptionMsg = ("negative score: " + nagetiveNum + " at index: " + i);
                                  throw new Exception(exceptionMsg);
                             }

                         }
               } catch (ClassCastException e){
                 // BUGFIX: not an Integer element in the Vector
//                      exceptionMsg = ("not an Integer element: " + sv.get(i).toString());
//                   System.err.println(exceptionMsg);
                     exceptionMsg = ("" + sv.remove(i));
                   System.err.println(e.toString() + ": " + exceptionMsg);
          } catch (Exception e) {
                   System.err.println(e.toString());
                   //e.printStackTrace();
          }
                switch (summaryMethod) {

                 case LAST_SCORE:
                        return (sv.lastElement());

                 case AVG_SCORE:
                  Iterator<Integer> si = sv.listIterator();
                        Integer sum = 0;
                         while (si.hasNext()) {
                           sum += si.next();
                        }
                        return ((sv.size() == 0) ? null : (sum / sv.size()));

                 default:
                 case MAX_SCORE:
                  return (Collections.max(sv));

                 }
        }

         public double scaledMark(Vector<Integer> sv){
//          quiz.summariseScores(sv);
              double temp = (double)this.summariseScores(sv) / maxScore * maxMarks;
            return temp;
     }
        public Quiz(String n, Course c, int ma, SummaryStatistic sstat) {
               name = n;
               //System.out.println(name);
             assigned = c;
            maxAttempts = ma;
               summaryMethod = sstat;
   }
}
```

cs230_ass2 / Asst2 / qwta / Scoresheet.java

```
package qwta;

import java.util.Vector;

public class Scoresheet {

         public Quiz scored;
      public Marksheet recorded;
       private Vector<Integer> scores;

         // String-valued utility function for reporting on the number of attempts
        private String attemptsAt() {
       if( scores.size() == 1) {
                return "1 attempt at ";
          } else {
                 return scores.size() + " attempts at ";
          }
     }

         public void addScore( int score ) {
              try {
                    // BUGFIX : reached attempts limit
```

```
                if (scores.size() > scored.maxAttempts)
                   throw new Exception("reached attempted limit: " + scores.size());
                 scores.add( score );
        } catch (Exception e){
              System.err.println(e.toString()/*+"\n"*/);
          }

    }

    public void reportOnScoresForStudent() {
            String outstr;
     outstr = "Report for student " + recorded.marked.name + ": ";
       outstr += "In " + (recorded.assessed).name + ", you made ";
         outstr += attemptsAt() + scored.name;
       outstr += ".  Marks = " + scored.scaledMark(scores)/*summariseScores( scores )*/;
           System.out.println( outstr );
}

    public String reportScoresToInstructor() {
          String retval;
      retval = "In " + recorded.assessed.name;
            retval += ", student " + recorded.marked.name + " made ";
           retval += attemptsAt() + scored.name;
       retval += ".  Marks = " + scored.scaledMark(scores)/*summariseScores( scores )*/;
           //System.out.println(retval);
       return retval;
}

    public Scoresheet(Quiz q, Marksheet m) {
                scored = q;
             recorded= m;
            scores = new Vector<Integer>();
 }

}
```

cs230_ass2 / Asst2 / qwta / **Student.java**

```
package qwta;

import java.util.Iterator;
import java.util.Vector;

public class Student {

  public String name;
  // public Vector<Course>  enrolled;  (implied: students can do this through Cecil)
  // public Vector<Marksheet>  marked;  (implied: students can do this through Cecil)
  public Vector<Scoresheet> sat; // Our student must remember the quizzes they have attempted!

  // QW probably should have a use-case to allow students to look up their scoresheets

  public void takeQuiz(Scoresheet ss) {
    ss.scored.sitQuiz( ss );
        sat.add( ss );
  }

  public void requestReport( ) {
     Iterator<Scoresheet> ssi = sat.listIterator();
          while( ssi.hasNext() ) {
                ssi.next().reportOnScoresForStudent();
          }
  }

  public Student(String n) {
    name = n;
    sat = new Vector<Scoresheet>();
  }

}
```