Programming in Java 2004 Paper 1 Question 10 (AFB)

This is a full executable solution. It will be possible to obtain full marks without including all detail here, but of course the main design components must all be present, and in appropriate syntax.

```
import java.io.*;
import java.util.*;

/**
 * Class of students for marking
 *
 * @author Alan Blackwell
 * @version 12 Feb 2004
 */

public class examClass
{
private ArrayList students;
private ArrayList questions;

public examClass()
{
    questions = new ArrayList();
    modelAnswer ma = new question1model();
    question q = new question(ma);

    students = new ArrayList();
    studentPaper sp = new studentPaper();

    sp.markQuestion(q, new answer());

    questions.add(q);
    students.add(sp);
}

public void reportResults()
{
    Iterator student_it = students.iterator();
    Iterator question_it = questions.iterator();
    while (student_it.hasNext()) {
        studentPaper s = (studentPaper)student_it.next();
        System.out.println("Student grade: " + s.grade());
    }
    while (question_it.hasNext()) {
        question q = (question)question_it.next();
        System.out.println("Question average: " + q.average());
```

```
    }
}
}


/**
 * A paper marked for a single student
 *
 * @author Alan Blackwell
 * @version 12 Feb 2004
 */

public class studentPaper
{
    // mark for a whole paper
    private int possibleTotal;
    private int actualTotal;

    /**
     * Constructor for students
     */
    public studentPaper()
    {
        // initialise instance variables
        possibleTotal = 0;
        actualTotal = 0;
    }

    /**
     * Mark a question completed by this student
     *
     * @param  q the question to be marked
     * @param  a the answer by this student
     * @return the mark out of 20
     */
    public int markQuestion(question q, answer a)
    {
        int m = q.mark(a);
        possibleTotal += 20;
        actualTotal += m;
        return m;
    }

    /**
     * Return percentage grade for the whole paper
     *
     * @return the percentage achieved in all questions completed
```

```
     */
     public int grade()
    {
        if (possibleTotal == 0) {
            return 0;
        } else {
            return (actualTotal * 100) / possibleTotal;
        }
    }
}


import java.io.*;
import java.net.URL;
import java.util.*;

/**
 * Class to contain a java exam answer
  *
 * @author Alan Blackwell
 * @version 12 Feb 2004
*/

public class answer
{
    // Store each line of Java as a string
    private ArrayList codeLines;
    // Count of number of correct lines
    private int correctLines;
    // Count expected lines that are missing
    private int missingLines;

    /**
     * Read from a default file for testing.
     */
    public answer()
    {
        this("sampleAnswer.txt");
    }

    /**
     * Read from an actual answer file
     * @param filename The file containing the answer
     */
    public answer(String filename)
    {
        // Locate the file with respect to the current environment.
```

```
    URL fileURL = getClass().getClassLoader().getResource(filename);
    // Where to store the data.
    codeLines = new ArrayList();

    // Attempt to read all lines from the file.
    try{
        BufferedReader answerfile =
            new BufferedReader(new FileReader(fileURL.getFile()));
        // Read the data lines until the end of file.
        String line = answerfile.readLine();
        while(line != null) {
            codeLines.add(line);
            line = answerfile.readLine();
        }
        answerfile.close();
    }
    // Note error processing not required - included here for testing
    catch(IOException e) {
        System.out.println("Failed to read the answer file: " + filename);
    }
    correctLines = 0;
    missingLines = 0;
}

public boolean testLine(String target)
{
    Iterator it = codeLines.iterator();
    while (it.hasNext()) {
        String candidate = (String) it.next();
        if (candidate.equals(target)) {
            foundLine(candidate);
            return true;
        }
    }
    missingLines++;
    return false;
}

/**
 * Once a line has been found, score and remove it
 */
public void foundLine(String line)
{
    correctLines++;
    codeLines.remove(line);
}
```

```
    public int calculateGrades()
    {
        float ratioCorrect = ((float)correctLines) / (correctLines + missingLines);
        return (int) (20.0 * ratioCorrect);
    }

}


/**
 * Mark scripts and maintain total marks for all answers to a question
 *
 * @author Alan Blackwell
 * @version 12 Feb 2004
 */

public class question
{
    // all marks for this question
    private int total;
    private int numberOfAnswers;
    private modelAnswer model;

    /**
     * Constructor for objects of class question
     */
    public question(modelAnswer ma)
    {
        model = ma;
        total = 0;
        numberOfAnswers = 0;
    }

    /**
     * Calculate average of collected marks
     *
     * @return     the average
     */
    public float average()
    {
        if (numberOfAnswers == 0) {
            return 0;
        } else {
            return total / numberOfAnswers;
        }
    }
```

```
    /**
     * Mark an answer to this question, comparing to model answer
     *
     * @param answer - the answer that is to be marked
     * @return mark out of 20
     */
     public int mark(answer a)
    {
        while (model.hasMoreLines()) {
            a.testLine(model.nextLine());
        }
        int result = a.calculateGrades();
        total += result;
        numberOfAnswers++;
        return result;
    }

}


import java.util.*;

/**
 * Abstract class modelAnswer - base class for all model answers
 *
 * @author Alan Blackwell
 * @version 12 Feb 2004
 */

public abstract class modelAnswer
{
// the lines required in the model answer
protected ArrayList expectedLines;
    // An iterator over entries.
    private Iterator dataIterator;

public modelAnswer()
{
    expectedLines = new ArrayList();
    defineAnswer();
    reset();
}

abstract protected void defineAnswer();

    /**
     * Set up a fresh iterator to traverse answer
```

```
     */
    public void reset()
    {
        dataIterator = expectedLines.iterator();
    }

public boolean hasMoreLines()
    {
        return dataIterator.hasNext();
    }

    public String nextLine()
    {
       return (String) dataIterator.next();
    }

}


/**
 * Model answer for question 1
 *
 * @author Alan Blackwell
 * @version 12 Feb 2004
 */

public class question1model extends modelAnswer
{
/**
 * The actual answer for this question
 */
protected void defineAnswer()
{
expectedLines.add("import java.io.*;");
expectedLines.add("public class helloWorld");
expectedLines.add("system.out.println(\"hello world\");");
}
}
```