This examination has 11 pages: check that you have a complete paper.
Check that you have a complete paper.
Each candidate should be prepared to produce, upon request, his or her SUNY/UB card.
This is a closed book, closed notes, closed neighbor examination.
Use of calculators, cell phones, mp3 players, or any aids whatsoever, is NOT permitted.
This examination has 10 questions.
Each question has a part worth 120 competency points and a part worth 1 proficiency point.
Answer all 10 questions.
You have 3 hours (180 minutes) to complete this examination.

READ AND OBSERVE THE FOLLOWING RULES:

- Write the following, in ink, in the space provided on the back page,
  - your usual signature, and
  - your UB person number,
- All of your writing must be handed in. This booklet must not be torn or mutilated in any way, and must not be taken from the examination room.
- Show all of your work in arriving at an answer, unless instructed otherwise. Partial credit will be awarded as appropriate.
- Candidates are not permitted to ask questions of the invigilators, except in cases of supposed errors or ambiguities in examination questions.
- **You may not leave the examination room in first 30 minutes of the exam, or in the last 15 minutes.**
- **You may not enter the examination room after the first 30 minutes of the exam.**
- CAUTION – Candidates guilty of any of the following, or similar, dishonest practices shall be **immediately dismissed** from the examination and shall be liable to disciplinary action.
  - Making use of any memory, computational, communication or similar devices, including but not limited to books, papers, calculators, computers, cell phones, and Google glass.
  - Speaking or communicating with other candidates, in any way.
  - Purposely exposing written papers to the view of other candidates

  The plea of accident or forgetfulness shall not be received.

DO NOT WRITE BELOW

Competency Points

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	COMPETENCY
/120	/120	/120	/120	/120	/120	/120	/120	/120	/120	/1200

Proficiency Points

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	PROFICIENCY
/1	/1	/1	/1	/1	/1	/1	/1	/1	/1	/10

Question 1 [Competency]

This question assesses your ability to write basic code elements.

Define a class which whose fully qualified name is `q1.EveryKeyListener`. *[GRADING: 2 points]*
The class must implement the `java.awt.event.KeyListener` interface, whose definition is given here:

```
package java.awt.event;
import java.util.EventListener;
public interface KeyListener extends EventListener {
    public void keyTyped(KeyEvent e);
    public void keyPressed(KeyEvent e);
    public void keyReleased(KeyEvent e);
}
```

The class must have:

1. An instance variable of type `code.Model` *[GRADING: 2 points]*
2. A constructor with one parameter whose value is used to initialize the instance variable. *[GRADING: 2 points]*
3. Definitions for all the methods of the `KeyListener` interface, as follows:
   a. The `keyTyped` method must call the method `remove` on the `code.Model` object whose reference is stored in the instance variable.
   Assume that `remove` method takes a `char` as an argument. Pass the value returned by `getKeyChar()` method (called on a `KeyEvent` object) as argument. *[GRADING: 2 points]*
   b. The `keyPressed` method must be defined so that it does nothing when called. *[GRADING: 1 point]*
   c. The `keyReleased` method must be defined so that it does nothing when called. *[GRADING: 1 point]*

Do NOT include anything else in the definition of the class. Write your answer below:

```
package q1;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import code.Model;
public class EveryKeyListener implements KeyListener {
        private Model _model;

        public EveryKeyListener(Model m) {
                _model = m;
        }
        @Override public void keyTyped(KeyEvent e) {
                _model.remove(e.getKeyChar());
        }
        @Override public void keyPressed(KeyEvent e) {}
        @Override public void keyReleased(KeyEvent e) {}
}
```

Question 1 [Proficiency]

Define an anonymous event handler for a JButton that uses System.out.println to print "click" whenever the button it is attached to is clicked.

```
button.addActionListener(new ActionListener() {
        @Override public void actionPerformed(ActionEvent e) { System.out.println("click"); }

});
```

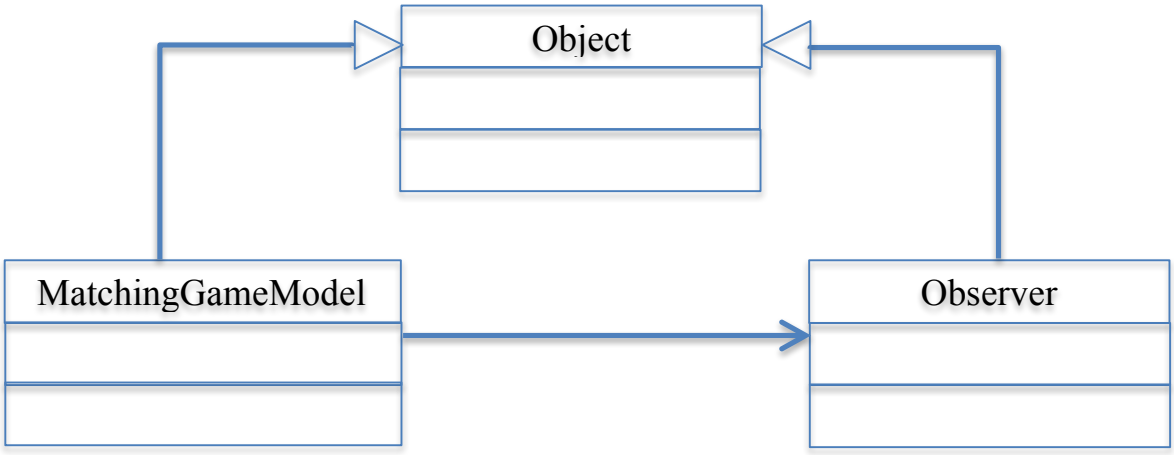
Question 2 [Competency] *This question assesses your command of UML class diagram notation.*

[GRADING: 120 total marks – 120 points for a correct diagram, 80 points for minor mistakes/omissions, 40 points for a diagram with more than minor mistakes/omissions, 0 points otherwise (e.g. no answer or wrong type of diagram)]

Draw a UML class diagram for the following code. Show all relationships present in the code (even if implicit).

```
package code.model;

public class MatchingGameModel {

        private Observer _observer;
       private Board _board;

        public MatchingGameModel() {
                _observer = null;
                _board = new Board(this);
        }

        public void setObserver(Observer obs) {
                _observer = obs;
        }

        // OTHER CODE OMITTED

}
```

If Observer is drawn as an interface (a box with just two compartments) and no relationship to Object, that is acceptable as well.



Question 2 [Proficiency]
What is the name of the design pattern used for event handling in Java.

It is the Observer Pattern.

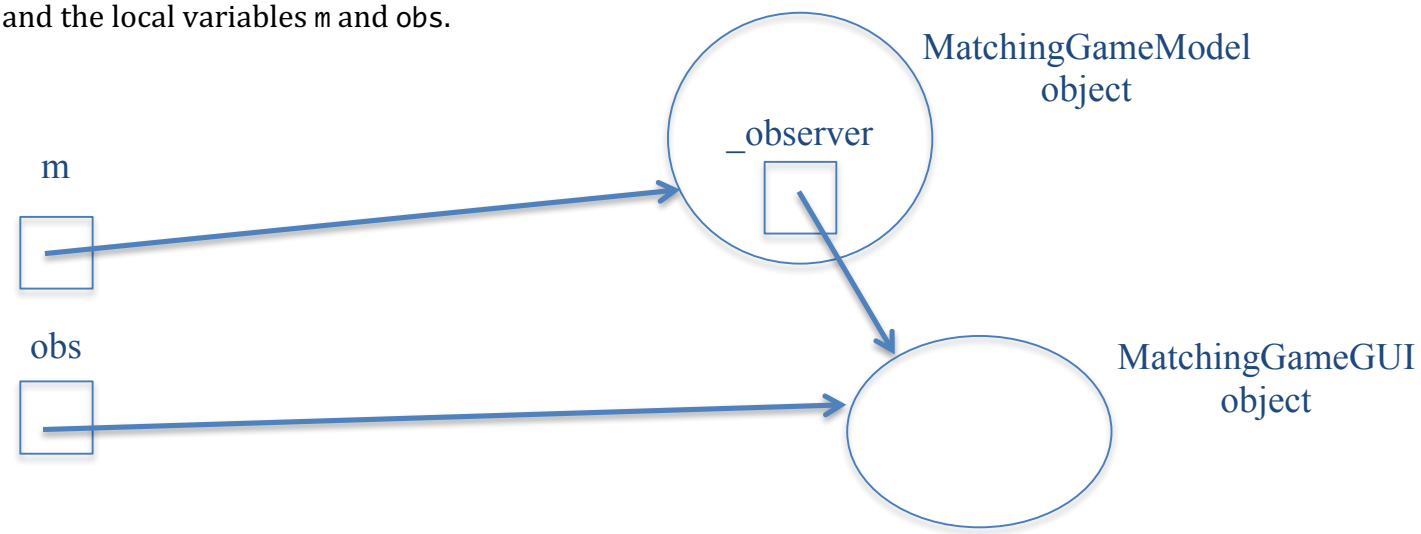
Question 3 [Competency] *This question assesses your understanding of variables, objects and references.*

[GRADING: 120 total marks – 120 points for a correct diagram, 80 points for minor mistakes/omissions, 40 points for a diagram with more than minor mistakes/omissions, 0 points otherwise (e.g. no answer or wrong type of diagram)]

Given the code in question 2, draw an object diagram showing the state of the program after three lines are executed (but before any memory reclamation happens):

```
MatchingGameModel m = new MatchingGameModel();
Observer obs = new MatchingGameGUI();
m.setObserver(obs);
```

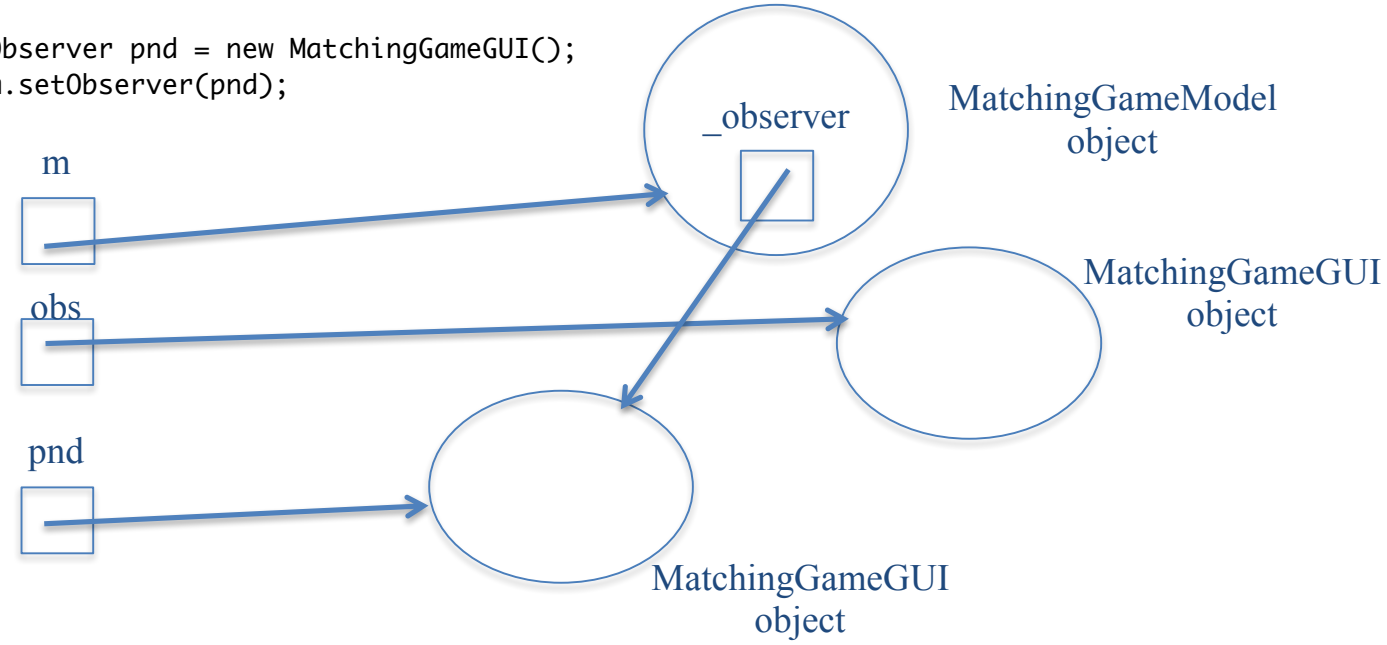
Show all objects explicitly created in the code, all instance variables explicitly declared in the code, and the local variables `m` and `obs`.



Question 3 [Proficiency]

Redraw the diagram assuming that also this code is executed after the above code is run:

```
Observer pnd = new MatchingGameGUI();
m.setObserver(pnd);
```



Question 4 [Competency]

This question assesses your basic vocabulary comprehension.

[GRADING: 120 total marks – 12 points for each correctly identified item.]

In the code below, clearly circle and identify by number **one (and only one)** example of each of the following. Be sure to identify each item precisely (circle no more and no less than necessary). If you believe no example exists in the given code, write "*no example*" next to that item.

1. an assignment operator
2. a relational operator
3. a parameter list
4. a for statement (entire statement)
5. the name of a supertype
6. a boolean literal value
7. a comment
8. an int expression
9. a local variable declaration
10. a method call

```
package code.gui;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GridLayout;
import javax.swing.JButton;
import javax.swing.JPanel;
import code.model.MatchingGameModel;
import code.model.Observer;

public class MatchingGameGUI implements Runnable, Observer {
        private MatchingGameModel _model;
        private JPanel _boardPanel;
        private JFrame _window;

        public MatchingGameGUI(MatchingGameModel m) {
          _model = m;
          _model.setObserver(this);
        }

        @Override public void run() { /* DEFINITION OMITTED */ }

        public void createAndPopulateBoard() {
          _boardPanel = new JPanel();
          _boardPanel.setFocusable(false);
          _boardPanel.setLayout(
            new GridLayout(MatchingGameModel.ROWS, MatchingGameModel.COLS));
          for (int c=0; c<MatchingGameModel.COLS; c++) {
            for (int r=0; r<MatchingGameModel.ROWS; r++) {
              JButton b = new JButton();
              b.setOpaque(true);
              b.setFocusable(false);
              b.setPreferredSize(new Dimension(100, 100));
              b.setFont(b.getFont().deriveFont(Font.BOLD, b.getFont().getSize()*4));
              _boardPanel.add(b);
            }
          }
          update();
        }
        @Override public void update(){ /* DEFINITION OMITTED */ }
}
```

Question 4 [Proficiency]

Identify a Java annotation in the above code.

Question 5 [Competency] *This question assesses your ability to write a simple method involving a loop.*

[GRADING: 120 total marks – 120 points for a correct method, 80 points for minor mistakes/omissions, 40 points for a method with more than minor mistakes/omissions, 0 points otherwise (e.g. no answer, no method, or a completely incorrect method)]

Define a `void` method named `write` which has a parameter of type `ArrayList<Beverage>`. You may name the parameter however you wish, as long as the name adheres to our naming conventions for parameters. Define the method so it prints the name, flavor and calorie count of each of the `Beverage` objects in the `ArrayList` on a separate line. The `Beverage` class is defined as shown below.

```
package q6;
public class Beverage {
        private String _name;
        private String _flavor;
        private int _calories;
        public Beverage(String n, String f, int c) {
                _name = n;
                _flavor = f;
                _calories = c;
        }
        public int calories() { return _calories; }
        public String name() { return _name; }
        public String flavor() { return _flavor; }
}
```

For example, running this code,

```
ArrayList<Beverage> list = new ArrayList<Beverage>();
list.add(new Beverage("H2O","nothing",0));
list.add(new Beverage("Ginger Ale","ginger",120));
list.add(new Beverage("Flavored tea","peach",15));
write(list);
```

must produce the following output:

```
H2O tastes of nothing and has 0 calories.
Ginger Ale tastes of ginger and has 120 calories.
Flavored tea tastes of peach and has 15 calories.
```

```
public void write(ArrayList<Beverage> list) {
        for (Beverage d : list) {
                System.out.println(d.name() + " tastes of " + d.flavor() +
                        " and has "+d.calories()+" calories.");
        }
}
```

Question 5 [Proficiency]

Rewrite the method to produce padded output so all names line up, based on the longest name in the list. For the above example the output must be:

```
H2O          tastes of nothing and has 0 calories.
Ginger Ale   tastes of ginger and has 120 calories.
Flavored tea tastes of peach and has 15 calories.
```

```
public static void write2(ArrayList<Beverage> list) {
        String longest = "";
        for (Beverage d : list) {
                while (d.name().length() > longest.length()) {
                        longest = longest + " ";
                }
        }
        for (Beverage d : list) {
                System.out.println(d.name() + longest.substring(0, longest.length()-d.name().length())+"
tastes of " + d.flavor() + " and has "+d.calories()+" calories.");
        }
}
```

Question 6 [Competency] *This question assesses your ability to write a simple method involving a conditional.*

[GRADING: 120 total marks – 120 points for a correct method, 80 points for minor mistakes/omissions, 40 points for a method with more than minor mistakes/omissions, 0 points otherwise (e.g. no answer, no method, or a completely incorrect method)]

Consider the following table mapping numeric grades to letter grades:

>=	<	Letter grade
	60	F
60	70	D
70	80	C
80	90	B
90		A

Define a method named `gradeMapper` which accepts as an argument a numeric grade (expressed as a value of type `int`) and returns the corresponding letter grade, according to the table above, as a `String`. You may assume that the argument will be in the range 0 to 100.

For example, `gradeMapper(81)` must return the `String` `"B"`.

```
public String gradeMapper(int ng) {
     if (ng < 60) { return "F"; }
     if (ng < 70) { return "D"; }
     if (ng < 80) { return "C"; }
     if (ng < 90) { return "B"; }
     return "A";
}
```

Question 6 [Proficiency]
Solve the same problem in a different way. For example, if you used if-else statements above, use if statements instead.

```
public String gradeMapper(int ng) {
     String answer = "A";
     if (ng < 60) { answer = "F"; }
     else if (ng < 70) { answer = "D"; }
     else if (ng < 80) { answer = "C"; }
     else if (ng < 90) { answer = "B"; }
     return answer;
}
```

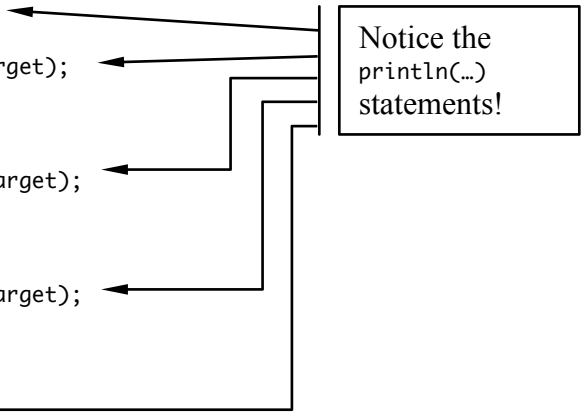
Question 7 [Competency]

This question assesses your code tracing/reading abilities.

[GRADING: 120 total marks – 120 points for a correct answer, 80 points for an answer which demonstrates an understanding of how the code executes, but which has some minor mistakes, 40 points for an answer that is relevant to question but clearly wrong, displaying minimal understanding of how the code executes, 0 points otherwise.]

Consider the following method:

```
public boolean binarySearch(ArrayList<Integer> list, int target) {
    int left = 0;  // first included element
    int right = list.size();  // first excluded element
    while ( left != right ) {
        int mid = (left + right) / 2;
        System.out.println("L: "+left+" M: "+mid+" R: "+right);
        if ( list.get( mid ) == target ) {
               System.out.println("---> "+list.get(mid)+"=="+target);
               return true;
        }
        if ( list.get( mid ) > target )  {
               System.out.println("---> "+list.get(mid)+" > "+target);
               right = mid;
        }
        if ( list.get( mid ) < target )  {
               System.out.println("---> "+list.get(mid)+" < "+target);
               left = mid + 1;
        }
    }
    System.out.println("Not found.");
    return false;
}
```



Show what is printed when the following code is executed:

```
ArrayList<Integer> list = new ArrayList<Integer>();
list.add(375);
list.add(402);
list.add(452);
list.add(473);
list.add(522);
list.add(578);
list.add(685);
list.add(792);
binarySearch(list, 620);
```

```
L: 0 M: 4 R: 8
---> 522 < 620
L: 5 M: 6 R: 8
---> 685 > 620
L: 5 M: 5 R: 6
---> 578 < 620
Not found.
```

Question 7 [Proficiency]

Assume each println statement was modified to include the value of mid at the front, as in this example:

```
System.out.println(mid + " ---> "+list.get(mid)+"=="+target);
```

What would be printed when running the above code?

```
L: 0 M: 4 R: 8
4---> 522 < 620
L: 5 M: 6 R: 8
6---> 685 > 620
L: 5 M: 5 R: 6
5---> 578 < 620
Not found.
```

Question 8 [Competency]

This question assesses your comprehension of inheritance, overriding and polymorphism.

[GRADING: 120 total marks – 120 points for a completely correct answer, 80 points for a response with minor mistakes/omissions, 40 points for an answer that is clearly wrong but somehow related to the question at hand, and 0 points otherwise.]

Consider the following definitions:

```
public abstract class Base {
        private int _v;
        private int _w;
        public Base() { this(2); }
        public Base(int x) { this(x,3); }
        public Base (int r, int s) { _v = r; _w = s; }
        public int foo() {
                return _v * _w;
        }
        public int v() { return _v; }
        public int w() { return _w; }
        public abstract String name();
}
public class Derived extends Base {
        public Derived() { super(5); }
        @Override public String name() { return "Derived"; }
}
public class Child extends Base {
        public Child() { super(4,7); }
        @Override public int foo() {
                return super.foo() + 3;
        }
        @Override public String name() { return "Child"; }
}
public class Kid extends Base {
        @Override public int foo() {
                return 3*v() + 5*w();
        }
        @Override public String name() { return "Kid"; }
}
```

Show what is printed when the following code executes:

```
ArrayList<Base> list = new ArrayList<Base>();
list.add(new Derived());
list.add(new Child());
list.add(new Kid());
for (Base x : list) {
        System.out.println(x.name() + " produces " + x.foo());
}
```

```
Derived produces 15
Child produces 31
Kid produces 21
```

Question 8 [Proficiency]

What is printed if the same code is run, but Kid contains this definition:

```
@Override public int v() { return 2 * super.v() - 1; }
```

```
Derived produces 15
Child produces 31
Kid produces 24
```

Question 9 [Competency]

This question assesses your understanding of basic programming concepts.

[GRADING: 120 total marks – 24 points for each correctly defined term, 12 points for a reasonable attempt, 0 points otherwise.]

Briefly explain what a variable is:

A variable is a storage location in memory; it has a name (in the high-level program), a location (address), a type, a value (contents), lifetime and scope. A 2-point answer must capture at least that a variable can hold a value.

Briefly explain what a reference is:

A reference is the address of a block of memory.

Briefly explain what an object is:

An object is an instance of a class. An object has properties (instance variables: things it knows) and behaviors (methods: things it can do).

Briefly explain what **`this`** is:

`this` is an implicit parameter of every (instance) method; it holds a reference to the object on which the method was invoked.

Briefly explain what **`null`** is:

`null` is a reference that does not refer to any object.

Question 9 [Proficiency]

Briefly explain what JSON is, and how it can be used.

JSON, short for JavaScript Object Notation, is a human-readable textual format for storing and transmitting data. It can be used to communicate data from one application to another, independent of the languages used to implement those applications.

Question 10 [Competency]

This question assesses your understanding of JSON

[GRADING: 120 total marks – 120 points for a completely correct answer, 80 points for a response with minor mistakes/omissions, 40 points for an answer that is clearly wrong but somehow related to the question at hand, and 0 points otherwise.]

What is printed when the following main method is called?

```
public static void main(String[] args){
      // The double quotes need to be escaped with the \ character. This tells
      // Java that we want the " character instead of ending the String.
      String jsonString = "[5,8,\"data\",\"elements\"]";
      JsonValue jsonValue = Json.parse(jsonString);
      JsonArray jsonArray = jsonValue.asArray();

      JsonValue index0 = jsonArray.get(0);
      int firstValue = index0.asInt();

      JsonValue index1 = jsonArray.get(1);
      int secondValue = index1.asInt();

      JsonValue index2 = jsonArray.get(2);
      String thirdValue = index2.asString();

      JsonValue index3 = jsonArray.get(3);
      String fourthValue = index3.asString();

      System.out.println("firstValue: " + firstValue);
      System.out.println("secondValue: " + secondValue);
      System.out.println("thirdValue: " + thirdValue);
      System.out.println("fourthValue: " + fourthValue);
}
```

Question 10 [Proficiency]

How would you construct a JSON object that stores a student's name, their person number, a semester identifier (e.g. F17) and the set of courses they are taking (e.g. CSE115, CSE191, ENG201, MTH141).

```
JsonObject student = new JsonObject();
student.add("name", "Sally Smith");
student.add("person number", 11223344);
student.add("semester", "F17");
JsonArray courses = new JsonArray();
courses.add("CSE115");
courses.add("CSE191");
courses.add("ENG201");
courses.add("MTH241");
student.add("courses", courses);
```