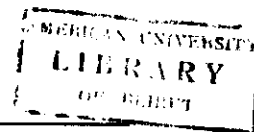




Faculty of Arts & Sciences
Department of Computer Science
CMPS 200—Introduction to Computer Programming
Fall 2004–2005
Thursday, January 27, 2005



Final Exam



Name: _____ Student Id: _____

Signature: _____ Section:

Lect I	10–11
Lect II	1–2

Duration: 120 minutes

General Instructions

- There are 2 parts and 22 pages. Make sure that you have all of them.
- The exam is closed book, closed notes, and closed neighbor.
- Please note that wrong answers to multiple choice questions carry a penalty: **four wrong answers cancel one correct answer.**
- Do not write anything besides your name on the pages with multiple choice questions. Write your answers on the separate answer sheet provided with Part II of the exam handout.
- Your handwriting should be readable so it can be graded.

Part I

Answer the following questions by circling the appropriate answer on the answer sheet.


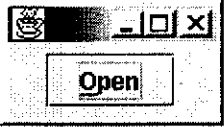
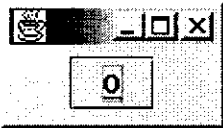
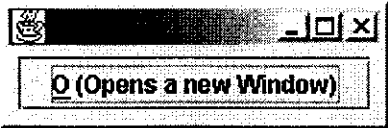

1	Q 09 8.5 (d)	<p>1. Consider the code below. Note that the catch statements in the code are not implemented, but you will not need those details. Assume filename is a String, x is an int, a is a double array and i is an int. Use the comments i1, i2, i3, e1, e2, e3, e4, e5 to answer the questions (i for instruction, e for exception handler).</p> <pre> try { BufferedReader infile = new BufferedReader(new FileReader(filename)); // i1 int x = Integer.parseInt(infile.readLine()); // i2 a[++i] = (double) (1 / x); // i3 } catch (FileNotFoundException ex) {...} // e1 catch (NumberFormatException ex) {...} // e2 catch (ArithmeticException ex) {...} // e3 catch (ArrayIndexOutOfBoundsException ex) {...} // e4 catch (IOException ex) {...} // e5 </pre> <p>An exception raised by the instruction in i3 would be caught by the catch statement labeled</p> <ul style="list-style-type: none"> a) e2 <input checked="" type="radio"/> b) e3 c) e4 d) either e3 or e4 e) either e2, e3 or e4
2	Q15 9.18 (d)	<p>2. Which of the following is a listener class for a JSlider? ?</p> <ul style="list-style-type: none"> a) ListListener b) ActionListener c) ButtonListener d) ChangeListener e) SlideListener
3	Q 07 7.23 (e)	<p>3. Assume that Student, Employee, and Retired are all extended classes of Person, and all four classes have different implementations of the method getMoney. Consider the following code where ... are the required parameters for the constructors:</p> <pre> Person p = new Person(...); int m1 = p.getMoney(); // assignment 1 p = new Student(...); int m2 = p.getMoney(); // assignment 2 if (m2 < 100000) p = new Employee(...); else if (m1 > 50000) p = new Retired(...); int m3 = p.getMoney(); // assignment 3 </pre> <p>The reference to getMoney() in assignment 3 is to the class</p> <ul style="list-style-type: none"> a) Person b) Student c) Employee d) Retired e) None of the above, this cannot be determined by examining the code

4	Q 10 8.8 (b)	<p>4. Which of the following messages passed to the <code>String str</code> could throw a <code>StringIndexOutOfBoundsException</code>?</p> <ul style="list-style-type: none"> a) <code>str.length()</code> b) <code>str.charAt(2);</code> c) <code>replace('a', 'A');</code> d) <code>str.equals(str);</code> e) Any of the above could throw a <code>StringIndexOutOfBoundsException</code>.
5	Q 07 7.27 (c)	<p>5. Consider the following class definition:</p> <pre>public class SomeClass { private int x; public SomeClass(int newValue) { x = newValue; } }</pre> <p>If <code>q1</code> and <code>q2</code> are objects of <code>SomeClass</code>, then <code>q1.equals(q2)</code></p> <ul style="list-style-type: none"> a) is a syntax error since <code>equals</code> is not defined in the <code>SomeClass</code> class b) is true if <code>q1</code> and <code>q2</code> both store the same value of <code>x</code> c) is true if <code>q1</code> and <code>q2</code> reference the same <code>SomeClass</code> object d) is never true e) throws a <code>NullPointerException</code>
6	Q 03 7.12 (b)	<p>6. Which of the following is not a method of the <code>Object</code> class?</p> <ul style="list-style-type: none"> a) <code>clone</code> b) <code>compareTo</code> c) <code>equals</code> d) <code>toString</code> e) All of the above are methods of the <code>Object</code> class.
7	Q 04 7.16 (c)	<p>7. Consider the following class definition:</p> <pre>public class AClass { protected int x; protected int y; public AClass(int a, int b) { x = a; y = b; } public int addEm() { return x + y; } public void changeEm() { x++; } }</pre>

		<pre> y--; } public String toString() { return "" + x + " " + y; } } </pre> <p>Which of the following would best redefine the toString method for BClass, a subclass of AClass?</p> <p>a) <pre>public String toString(int z) { return " " + x + " " + y + " " + z; }</pre></p> <p>b) <pre>public String toString() { return super.toString(); }</pre></p> <p>c) <pre>public String toString() { return super.toString() + " " + z; }</pre></p> <p>d) <pre>public String toString() { return super.toString() + " " x + " " + y + " " + z; }</pre></p> <p>e) <pre>public String toString() { return " " + x + " + y + " " + z; }</pre></p>
8	Q 01 7.1 (b)	<p>8. Consider the following partial class definitions:</p> <pre> public class A1 { public int x; private int y; protected int z; ... } public class A2 extends A1 { protected int a; private int b; ... } public class A3 extends A2 { private int q; ... } </pre> <p>Which of the following is true with respect to A1, A2 and A3?</p> <p>a) A1 is a subclass of A2 and A2 is a subclass of A3</p>

		<ul style="list-style-type: none"> b) A3 is a subclass of A2 and A2 is a subclass of A1 c) A1 and A2 are both subclasses of A3 d) A2 and A3 are both subclasses of A1 e) A1, A2 and A3 are all subclasses of the class A
9	Q14 9.10 (c)	<p>9. In order to allow the user to activate a JButton using the keyboard, you would add a _____ to the JButton.</p> <ul style="list-style-type: none"> a) shortcut b) KeyListener c) mnemonic d) ActionListener e) RadioButton
10	Q 02 7.9 (b)	<p>10. Java does not support multiple inheritance, but some of the abilities of multiple inheritance are available by</p> <ul style="list-style-type: none"> a) importing classes b) implementing interfaces c) overriding parent class methods d) creating aliases e) using public rather than protected or private modifiers
11	Q21 11.11 (c)	<p>11. Assume that <code>int[] a = {6, 2, 4, 6, 2, 1, 6, 2, 5}</code> and consider the two recursive methods below <code>foo</code> and <code>bar</code>.</p> <pre> public int foo(int[] a, int b, int j) { if (j < a.length) if (a[j] != b) return foo (a, b, j+1); else return foo (a, b, j+1) + 1; else return 0; } public int bar(int[] a, int j) { if (j < a.length) return a[j] + bar(a, j+1); else return 0; } </pre> <p>What is the result of calling <code>foo(a, 2, 0)</code> ;?</p> <ul style="list-style-type: none"> a) 0 b) 1 c) 2 d) 3 e) 4
12	Q17 11.1 (c)	<p>12. Consider the following recursive method.</p> <pre> public int questionA(int x, int y) { if (x == y) return 0; else return questionA(x-1, y) + 1; } </pre> <p>If the method is called as <code>questionA(8, 3)</code>, what is returned?</p> <ul style="list-style-type: none"> a) 11

		<p>b) 8 c) 5 d) 3 e) 24</p>
13	Q20 11.6 (c)	<p>13. Consider the following recursive factorial method:</p> <pre>public int factorial(int x) { if (x > 1) return x * factorial (x - 1); else return 1; }</pre> <p>How many times is the factorial method invoked if originally called with factorial(5)? Include the original method call in your counting.</p> <p>a) 1 b) 4 c) 5 d) 6 e) 7</p>
14	Q23 11.21 (d)	<p>14. Which of the following methods would properly compute the value of $x \% y$ ($x \bmod y$) recursively, assuming that x and y not negative numbers?</p> <p>a) <pre>public int recursiveMod(int x, int y) { if (x < y) return y; else return recursiveMod(x, y - x); }</pre></p> <p>b) <pre>public int recursiveMod(int x, int y) { if (x == y) return 0; else return recursiveMod(x - y, y) + 1; }</pre></p> <p>c) <pre>public int recursiveMod(int x, int y) { if (x < y) return x; else return recursiveMod(x - y, y) + 1; }</pre></p> <p>d) <pre>public int recursiveMod(int x, int y) { if (x < y) return x; else return recursiveMod(x - y, y); }</pre></p> <p>e) <pre>public int recursiveMod(int x, int y) { while (x > y) x = x - y; return x; }</pre></p>
15	Q13 9.9 (e)	<p>15. Assume JButton jbl has already been instantiated. To add a pop-up GUI box that will display "Closes the GUI" whenever the mouse is moved to the button and left there, you would do</p> <p>a) <code>jbl.add("Closes the GUI");</code> b) <code>jbl.setCaption("Closes the GUI");</code></p>

		<p>c) <code>jb1.addPopUpMessage("Closes the GUI");</code> d) <code>jb1.setMnemonic("Closes the GUI");</code> e) <code>jb1.setToolTipText("Closes the GUI");</code></p>
16	Q16 9.24 (a)	<p>16. The following code would be used to create which of the JButtons below?</p> <pre> JButton jb = new JButton ("Open"); jb.setMnemonic ('O'); jb.setToolTipText ("Opens a new Window"); </pre> <p>a) </p> <p>b) </p> <p>c) </p> <p>d) </p> <p>e) </p>
17	Q 11 8.12 (e)	<p>17. Consider the following skeletal code.</p> <pre> public static void main(String[] args) { try { ExceptionThrowerCode etc = new ExceptionThrowerCode (); etc.m1 (); etc.m2 (); } catch (ArithmeticException ae) { ... } } public class ExceptionThrowerCode { ... public void m1 () { ... } public void m2 () { try { m3 (); } catch(ArithmeticException ae) {...} catch(NullPointerException npe) {...} } public void m3 () { </pre>



		<pre> try { ... } catch (ArithmeticException ae) {...} } </pre> <p>If a <code>NullPointerException</code> arises in the try statement in m1,</p> <ol style="list-style-type: none"> it is caught in main it is caught in m1 it is caught in m2 it is caught in m3 it is not caught leading to the program terminating
18	Q19 11.3 (c)	<p>18. The following method should return true if the int parameter is even and either positive or 0, and false otherwise. Which set of code should you use to replace ... so that the method works appropriately?</p> <pre> public boolean question3(int x) { ... } </pre> <ol style="list-style-type: none"> if (x == 0) return true; else if (x < 0) return false; else return question3(x - 1); if (x == 0) return false; else if (x < 0) return true; else return question3(x - 1); if (x == 0) return true; else if (x < 0) return false; else return question3(x - 2); if (x == 0) return false; else if (x < 0) return true; else return question3(x - 2); return(x == 0);
19	Q18 11.2 (d)	<p>19. Consider the following recursive method.</p> <pre> public int question1_2(int x, int y) { if (x == y) return 0; else return question1_2(x-1, y) + 1; } </pre> <p>Calling this method will result in infinite recursion if which condition below is initially true?</p> <ol style="list-style-type: none"> (x == y) (x != y) (x > y) (x < y) (x == 0 && y != 0)
20	Q 08 8.3 (d)	<p>20. Consider the code below. Note that the catch statements in the code are not implemented, but you will not need those details. Assume filename is a String, x is an int, a is a double array and i is an int. Use the comments i1, i2, i3, e1, e2, e3, e4, e5 to answer the questions (i for instruction, e for exception handler).</p> <pre> try { BufferedReader infile = new BufferedReader(new FileReader(filename)); // i1 int x = Integer.parseInt(infile.readLine()); // i2 a[++i] = (double) (1 / x); // i3 } catch (FileNotFoundException ex) {...} // e1 catch (NumberFormatException ex) {...} // e2 catch (ArithmeticException ex) {...} // e3 catch (ArrayIndexOutOfBoundsException ex) {...} // e4 </pre>

		<pre>catch (IOException ex) {...} // e5</pre> <p>An exception raised by the instruction in i1 would be caught by the catch statement labeled</p> <ol style="list-style-type: none"> e1 e2 e5 either e1 or e5 either e1, e4 or e5
21	Q25 (d)	<p>21. Assume that a1 and a2 are arrays of length n. Which of the following pieces of code correctly determines the equality of all elements in the arrays:</p> <ol style="list-style-type: none"> <pre>boolean result = true; for (int i = 0; i < a1.length; i++) result = result && (a1[i] == a2[i]);</pre> <pre>boolean result = true; for (int i = 0; i < a1.length && result; i++) result = result && (a1[i] == a2[i]);</pre> <pre>boolean result = true; for (int i = 0; i < a1.length; i++) result = a1[i] == a2[i]</pre> <pre>boolean result = true; for (int i = 0; i < a1.length; i++) if (a1[i] == a2[i]) result = true; else result = false;</pre> <pre>boolean result = true; for (int i = 0; i < a1.length; i++) if (a1[i] != a2[i]) result = false;</pre> <ol style="list-style-type: none"> 1 2 3 & 4 1, 2, 5 All of the above.
22	Q22 11.17 (c)	<p>22. Why is the following method one which has infinite recursion?</p> <pre>public int infiniteRecursion(int n) { if (n > 0) return infiniteRecursion(n) + 1; else return 0; }</pre> <ol style="list-style-type: none"> Because there is no base case Because the base case will never be true Because the recursive call does not move the parameter closer to the base case Because the recursive call moves the problem further away from the base case None of the above, there is no infinite recursion in this method
23	Q 05 7.19 (d)	<p>23. A variable declared to be of one class can later reference an extended class of that class. This variable is known as</p> <ol style="list-style-type: none"> protected

		<ul style="list-style-type: none"> b) derivable c) cloneable d) polymorphic e) none of the above, a variable declared to be of one class can never reference any other type of class, even an extended class
24	Q 12 8.24 (a)	<p>24. To implement the <code>KeyListener</code> interface, you must implement all of the following methods except for which one?</p> <ul style="list-style-type: none"> a) <code>keyEvent</code> b) <code>keyPressed</code> c) <code>keyTyped</code> d) <code>keyReleased</code> e) all of the above methods must be implemented
25	Q24 (e)	<p>25. Assume that the method <code>booleanMeth</code> returns a <code>boolean</code> and takes an object reference as a parameter. Which of the following pieces of code uses types correctly:</p> <ol style="list-style-type: none"> 1. <code>if (booleanMeth(someObject)) System.out.println("booleanMeth: Object");</code> 2. <code>if (booleanMeth(someObject) == true) System.out.println("booleanMeth: Object true");</code> 3. <code>if (! booleanMeth(someObject)) System.out.println("NOT booleanMeth: Object");</code> 4. <code>if (booleanMeth(someObject) == false) System.out.println("booleanMeth: Object false");</code> 5. <code>if (booleanMeth(someObject) != true) System.out.println("booleanMeth: not equal Object");</code> <ul style="list-style-type: none"> a) 2 & 4 b) 1 c) 3 d) 5 e) All of the above.



Final Exam

Version 3

Name: _____ Student Id: _____

Signature: _____ Section:

Lect I	10–11
Lect II	1–2

Answers to Part I

Question	A	B	C	D	E
1.					
2.					
3.					
4.					
5.					
6.					
7.					
8.					
9.					
10.					
11.					
12.					
13.					
14.					
15.					
16.					
17.					
18.					
19.					
20.					
21.					
22.					
23.					
24.					
25.					

Grades

Part I	50	
Correct		
Incorrect		
Blank		
Part II	68	
1.1	20	
2.1	6	
2.2	6	
2.3	6	
2.4	4	
2.5	8	
3.1	6	
3.2	6	
3.3	6	
Total	118	

Part II—Programming

Problem 1—Binary Search

The *binary search* algorithm is based on the principle of successive approximation. It assumes that the list to be searched is sorted. It involves repeatedly dividing this list in half and deciding which half to look in next. The algorithm repeatedly divides the search list in half until the item is found or it is determined that the item is not in the list. More formally, given a sorted list specified by the first and last indices in a larger list, the algorithm searches through the list by (1) comparing the item being searched for with the middle element of the list, (2) deciding if the algorithm needs to search further, and, if yes, which half to search further—the bottom or the upper half, and (3) redefining the list to be searched as the identified half. This last step can be implemented by iteration or by recursion.

Example 1—Consider the example of searching for the number 106 in the following sorted list of 11 integers {12, 64, 72, 86, 92, 103, 106, 125, 200, 300, 400}. A walkthrough the algorithm yields the following steps:

1. The list's first and last indices are 0 and 10. The list's midpoint index is 5 and the midpoint number is 103. After comparing 106 and 103, the algorithm narrows down the search to the upper half of the current list—between indices 6 and 10.
2. The list's first and last indices are 6 and 10. The list's midpoint index is 8 and the midpoint number is 200. After comparing 106 and 200, the algorithm narrows down the search to the lower half of the current list—between indices 6 and 7.
3. The list's first and last indices are 6 and 7. The list's midpoint index is 6 and the midpoint number is 106. After comparing 106 and 106, the algorithm determines that the search is complete; the number is found at index 6.

Example 2—Consider searching for the number 24 in the same sorted list. A walkthrough of the algorithm yields the following steps:

1. The list's first and last indices are 0 and 10. The list's midpoint index is 5 and the midpoint number is 103. After comparing 24 and 103, the algorithm narrows down the search to the bottom half of the current list—between indices 0 and 4.
2. The list's first and last indices are 0 and 4. The list's midpoint index is 2; the midpoint number is 72. After comparing 24 and 72, the algorithm narrows down the search to the bottom half of the current list—between indices 0 and 1.
3. The list's first and last indices are 0 and 1. The list's midpoint index is 0; the midpoint number is 12. After comparing 24 and 12, the algorithm narrows down the search to the upper half of the current list—between indices 1 and 1.
4. In this step, the list has degenerated into a single number—the one at index 1 in the original list, namely 64. The algorithm compares 24 and 64 and concludes that the number 24 is not in the list.

This problem develops iterative and recursive versions of the binary search algorithm. The `BinarySearch` class will implement these iterative and recursive versions of the algorithm. The following driver class, `BinarySearchDriver`, illustrates the use of these two methods.

```
import java.util.Random;

public class BinarySearchDriver {
    public static void main(String[] args) {
        // sorted list
        int[] list = {12, 64, 72, 86, 92, 103, 106, 125, 200, 300, 400};
        Random random = new Random();
        int key = random.nextInt();

        int i1 = BinarySearch.iterativeSearch(key, list, 0, list.length);
        int i2 = BinarySearch.recursiveSearch(key, list, 0, list.length);

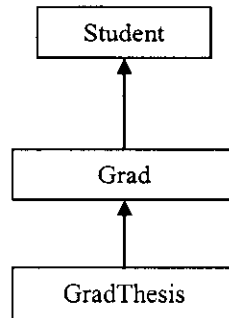
        if (i1 < 0) System.out.println("Iterative: did not find " + key);
        else System.out.println("Iterative: found " + key + " at " + i1);

        if (i2 < 0) System.out.println("Recursive: did not find " + key);
        else System.out.println("Recursive: found " + key + " at " + i2);
    }
}
```


Problem 2—Student Body

This programming exercise explores a class hierarchy that uses polymorphism via inheritance. Specifically, it explores a `StudentBody` class which represents various categories of `Student` objects. The first level of inheritance classifies one category of students as graduates. Graduates are all supported by a certain percentage of graduate assistantship but are further distinguished as being either writing a thesis or not writing a thesis.

Among many others, one objective of this program is to output information on the number of credit hours a given student is allowed to register for during a semester. This exercise implements the class hierarchy illustrated below.



Question 2.1—Student Class

This class is an abstract class which does not represent a particular type of student and is not intended to be instantiated. It serves as the ancestor of all student classes and contains information that applies to all students.

- a. Each student has a name (`String`) and a 6 digit ID number (`int`). The constructor of this class should accept these values as input and store them in variables, `name` and `id`, which are in turn inherited by all descendants of this class.
- b. This class overrides the `toString` method to return the information managed by the `Student` class.
- c. It finally contains an abstract method called `getCredits` which takes no parameters and returns a value of type `int`.

```
// Class header
```

```
{
```

```
public Student (                                     )  
{
```

```
    }  
    public String toString ()  
    {
```

```
    }
```

```
    // Additional methods
```

```
}
```

Question 2.2—Grad Class

The Grad class represents a graduate student that gets assigned a number of credits based on a formula that depends on *the percentage* of graduate assistantship the student has. This class is a descendant of the Student class.

- The constructor of this class accepts as input an integer representing the percentage of assistantship being offered, which is passed along with the other basic information to the Grad constructor, and is stored in the instance variable `percent_assist`.
- The `toString` method of this class overrides the inherited method to concatenate the additional information that Student introduces.
- The `getCredits` method of this class returns the number of credits as specified by the formula: $percentage * 10$.

```
public class Grad  
{
```

```
    public Grad (                                     )  
    {
```

```
    }  
    public String toString ()  
    {
```

```
    }  
    public int getCredits ()  
    {
```

```
    }  
}
```

Question 2.3—GradThesis Class

The `GradThesis` class represents a student who is not only a graduate student but also one who is writing a thesis (as indicated by a `boolean` named `thesis`). Such a student is first assigned a number of credits as it would for any graduate, then receives an extra 10 credits. This class is a descendant of the `Grad` class.

- The constructor of `GradThesis` takes no input parameters, passes along its information to the `Grad` constructor and initializes `thesis` to `false`.
- The `assignThesis` method assigns `thesis` according to a `boolean` input parameter it receives indicating that the given student is currently writing a thesis.
- The `toString` method of this class overrides the inherited method to concatenate the additional information that `GradThesis` introduces.
- The `getCredits` method of this class returns the number of credits as specified above.

```
public class GradThesis
{

    public GradThesis (                               )
    {

    }

    public void assignThesis(boolean thesis)
    {

    }

    public String toString ()
    {

    }

    public int getCredits ()
    {

    }

}
```

Question 2.4—University Class

This class contains the main driver that creates a `StudentBody` object and invokes the `credits` method to compute the number of credits to which any given student in `StudentBody` is entitled to.

```
public class University {
```

```
}
```

Question 2.5—StudentBody Class

This class maintains an array, `students`, of `Student` objects that represent four individual students, under the names of Diane and Norm, and of the following types:

- a. Diane, a graduate student with 100% Graduate assistantship not writing a thesis. ID number = 011639.
- b. Norm, a part-time graduate student with 50% Graduate assistantship and writing a thesis, ID number = 015689.

In the implementation of this class, check the specifications of the constructors of the various subclasses of `Student` before completing the following:

- a. The constructor method of the `StudentBody` class should declare the array to hold the 2 `Student` references as described in the categorization above. Only for the graduate students writing a thesis, the constructor should call the `assignThesis` method with a `true` parameter of the `GradThesis` class to indicate that the student is currently writing a thesis.
- b. The `credits` method should scan through the list of students, printing their names and ID numbers, and invoking their `getCredits` methods to output the number of credits each student is entitled to. The invocation of the `getCredits` method should be polymorphic as each class representing the various student categories has its own version of the `getCredits` method.

A sample output of this method should read:

- c. Norm, a part-time graduate student with 50% Graduate assistantship and writing a thesis, ID number = 015689, is entitled to 5 credits.

```
public class StudentBody {  
    private Student[] students;
```

```
    public StudentBody ()  
    {  
        students = new Student[2];
```

```
}  
public void credits ()  
{
```

```
}  
}
```

Problem 3—Set Inclusion

In this programming exercise, we will process a created exception based on a test for *set inclusion*. A set A is a subset of another set B if and only if all elements of A are found in B . Given two lists of integers, one shorter than the other, this program treats each list as a set and tests whether the first is a subset of the second. If not, an exception is declared and treated.

Question 3.1—SubsetTest Class

This class contains the main driver that

- Prompts the user to enter two lists of integers, one of which is shorter than the other. Before entering each list, the user is first prompted to enter its size. The integers in one list are read as a line of text with entries separated by white space. The text must be tokenized to extract individual list entries.
- Stores the two lists in two one-dimensional arrays, A and B .
- Instantiates an `ExceptionDemo` object which will demonstrate the process of creating a new exception based on the set inclusion test. Check the constructor of `ExceptionDemo` below.
- Invokes the method `level1` of the `ExceptionDemo` class to start processing the set inclusion test.

```
public class SubsetTest {
```

```
}
```

Question 3.2—ExceptionDemo Class

This class contains the following methods:

- This class's constructor accepts as input two one-dimensional arrays which represent two sets of integers.
- The method `level1` calls the method `level2` and is ONLY concerned with handling the created exception using the `getMessage` and `printStackTrace` methods.
- The method `level2` performs a subset test operation (whether A is a subset of B), generates and throws `NotSubsetException` if the result to this test is false.

```
public class ExceptionDemo
{
```

```
}
```

Question 3.3—NotSubsetException Class

This class represents an exceptional condition in which a given set is not a subset of the other. The constructor of this class should set the appropriate text message that the user wishes to display when an exception is detected.

```
public class NotSubsetException
```

```
{
```

```
}
```