

Name _____ Period _____

1. Refer to the code below,

```
public interface Sports {  
  
    void method1( );  
    void method2( );  
    int method3(double d);  
}  
  
public class Baseball implements Sports {  
  
    public Baseball( ) { . . . }  
    public void method1( ) { //some code...}  
    public void method2( ) { //some code...}  
    public int method3(double c ) { //some code...}  
    public int statevar1;  
}  
  
public class Football implements Sports {  
  
    public Football( ) { . . . }  
    public void method1( ) { //some code...}  
    public void method2( ) { //some code...}  
    public int method3(double c ) { //some code...}  
    public int statevar1;  
}  
  
public class Tester {  
  
    public static void main(String[] args) {  
  
        Sports x = new Baseball( );  
        Sports y = new Football( );  
        x.method2( );  
        y.method2( );  
        . . . more code . . .  
    }  
}
```

(a) Which methods, if any, in the Sports interface are abstract?

/1

(b) public class Hockey implements Sports {

//What methods, if any, must we implement here?

}

/1

(c) Look at the classes Baseball and Football. Both implement method1. Do both implementations have to have identical code? If so, why?	/1
(d) In the “more code” section of Tester what would the following return? (x instanceof Sports)	/1
(e) In the “more code” section of Tester what would the following return? (y instanceof Football)	/1
(f) The property of two classes being able to have methods of the same name (but with possibly different implementations) is known as	/1
(g) Modify the following class so that it will simultaneously inherit the Red class and implement both the Eagle and Bobcat interfaces. public class Austria { . . . }	/1

This question involves reasoning about 2-dimensional arrays of integers.

```
public class ArrayTester
{

    /** Returns an array containing the elements of column c of arr2D in the
    same order as
    * they appear in arr2D.
    * Precondition: c is a valid column index in arr2D.
    * Postcondition: arr2D is unchanged.
    */
    public static int[] getColumn(int[][] arr2D, int c)
    { /* to be implemented in part (a) */ }

    /** Returns true if and only if every value in arr1 appears in arr2.
    * Precondition: arr1 and arr2 have the same length.
    * Postcondition: arr1 and arr2 are unchanged.
    */
    public static boolean hasAllValues(int[] arr1, int[] arr2)
```

```
{ /* to be implemented in part (b) */ }
```

```
/** Returns true if arr contains any duplicate values;  
 * false otherwise.  
 */  
public static boolean containsDuplicates(int[] arr)  
{ /* to be implemented in part (c) */ }
```

```
/** Returns true if square is a Latin square as described in part (b);  
 * false otherwise.  
 * Precondition: square has an equal number of rows and columns.  
 * square has at least one row.  
 */  
public static boolean isLatin(int[][] square)  
{ /* to be implemented in part (d) */ }
```

- (a) Write a static method `getColumn`, which returns a one-dimensional array containing the elements of a single column in a two-dimensional array. The elements in the returned array should be in the same order as they appear in the given column.

The notation `arr2D[r][c]` represents the array element at row `r` and column `c`. The following code segment initializes an array and calls the `getColumn` method.

```
int[][] arr2D = {  
    { 0, 1, 2 },  
    { 3, 4, 5 },  
    { 6, 7, 8 },  
    { 9, 5, 3 }  
};  
  
int[] result = ArrayTester.getColumn(arr2D, 1);
```

When the code segment has completed execution, the variable `result` will have the following contents.
result: {1, 4, 7, 5}

/4

- (b) Write the static method `hasAllValues` which returns true if two arrays contain the same values. The values can appear in any order in either array.

/4

- (c) Write the static method `containsDuplicates` which returns true if an array has any duplicate values.

/4

- (d) Write the static method `isLatin`, which returns true if a given two-dimensional square array is a Latin square, and otherwise, returns false. In the `isLatin` method apply the `getColumn`, `hasAllValues`, and `containsDuplicates` helper methods.

A two-dimensional square array of integers is a Latin square if the following conditions are true.

- The first row has no duplicate values.
- All values in the first row of the square appear in each row of the square.
- All values in the first row of the square appear in each column of the square.

Examples of Latin Squares

1	2	3
2	3	1
3	1	2

10	30	20	0
0	20	30	10
30	0	10	20
20	10	0	30

Examples that are NOT Latin Squares

1	2	1
2	1	1
1	1	2

Not a Latin square
because the first row
contains duplicate
values

1	2	3
3	1	2
7	8	9

Not a Latin square
because the elements of
the first row do not all
appear in the third row

1	2
1	2

Not a Latin square
because the elements of
the first row do not all
appear in either column

