

# WES-CS GROUP MEETING #12

## Exercise 1: Practice with Inheritance

Below are six class definitions (ParentThing, ThingOne, ThingTwo, ThingThree, ChildOne, and ChildTwo). Remember that “extends” indicates inheritance. For instance, “ThingOne extends ParentThing” means that ThingOne is a subclass of ParentThing, and so ThingOne has all the data members and methods of ParentThing, plus some of its own.

```
class ParentThing {
    public int j;
    protected int k;
    private int p;

    public void methodA() { // Code segment 1 }
    public void methodA(String str) { // Code segment 2 }
    public void methodB() { // Code segment 3 }
}

class ThingOne extends ParentThing {
    public int onej;
    protected int onek;
    private int onep;

    public void methodA(int c) { // Code segment 4 }
}

class ThingTwo extends ParentThing {
    public void methodB() { // Code segment 5 }
    public void methodB(String str) { // Code segment 6 }
}

class ThingThree extends ParentThing {
    public void methodB() { // Code segment 7 }
}

class ChildOne extends ThingOne {
    public void methodA() { // Code segment 8 }
}

class ChildTwo extends ThingTwo {
    public void methodA() { // Code segment 9 }
}
```

Assume that the following statements have been executed.

```
ParentThing parent = new ParentThing();
ThingOne one = new ThingOne();
ThingTwo two = new ThingTwo();
ThingThree three = new ThingThree();
ChildOne kid1 = new ChildOne();
ChildTwo kid2 = new ChildTwo();
```

Divide into two teams to play a game like Jeopardy with 100, 200, and 300-point questions about the code above and on the previous page. For each question, the team that rings the bell first gets to try to answer the question. If the answer is right, the team gets points, and gets to choose the point value of the next question.

Some questions involve method calls made from *outside* the six classes (using the variables defined above). For each, you must say whether the code compiles without error, and if it does compile, which code segment in the class definitions given on the previous page executes when the method is called.

Other questions involve replacements for the code segments labeled

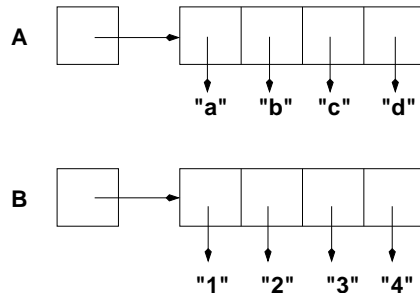
```
// Code segment 1
// Code segment 2
```

etc. For each, you must say whether the statement will cause a compile-time error and if so why.

The questions and answers are at the end of this set of exercises.

## Exercise 2: Practice with Arrays

Assume that you have two arrays, A and B, initialized as shown below.

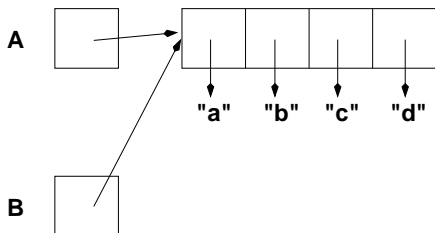


Here is an incomplete method that has two array parameters:

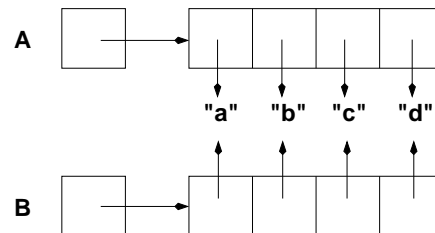
```
public static void change( String A[], String B[] ) {
    // missing code
}
```

For each of the pictures shown below, write code for the body of the `change` method so that calling `change(A, B)` would change arrays A and B from their initial values as shown above, to the new value shown in the picture. If there is no such code, explain why.

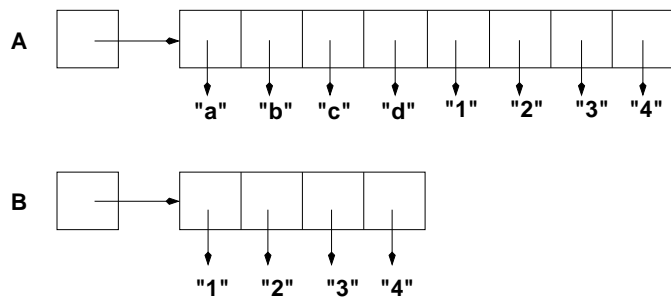
**Picture 1**



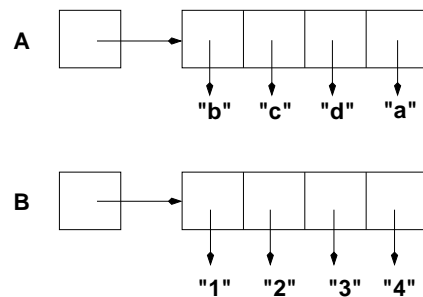
**Picture 2**



**Picture 3**



**Picture 4**



Now write code that would change array A:

<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>
<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>

to each of the following (your code should work for a rectangular two-dimensional array of any size):

<b>4</b>	<b>1</b>	<b>2</b>	<b>3</b>
<b>8</b>	<b>5</b>	<b>6</b>	<b>7</b>
<b>12</b>	<b>9</b>	<b>10</b>	<b>11</b>

<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>
<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>

### Exercise 3: The Lights-Out Game

For this exercise you will program a game for one player. We have a grid of lighted buttons, and the aim is to turn all the lights off. It's not as easy as turning each button off in turn because pressing a button changes its on/off state *and* as the states of its immediate horizontal and vertical neighbors.

Below is an example to show how your program should work for a 3-by-3 grid; your program would actually print the grids one under the other, but to save space they're shown here one next to the other.

```
 1,on | 2,on | 3,on
-----+-----+-----
 4,on | 5,on | 6,on
-----+-----+-----
 7,on | 8,on | 9,on
```

Enter position: 1

```
 1,off | 2,off | 3,on
-----+-----+-----
 4,off | 5,on | 6,on
-----+-----+-----
 7,on | 8,on | 9,on
```

Enter position: 5

---

```
 1,off | 2,on | 3,on
-----+-----+-----
 4,on | 5,off | 6,off
-----+-----+-----
 7,on | 8,off | 9,on
```

Enter position: 2

```
 1,on | 2,off | 3,off
-----+-----+-----
 4,on | 5,on | 6,off
-----+-----+-----
 7,on | 8,off | 9,on
```

Enter position: 1

---

```
 1,off | 2,on | 3,off
-----+-----+-----
 4,off | 5,on | 6,off
-----+-----+-----
 7,on | 8,off | 9,on
```

Enter position: 7

```
 1,off | 2,on | 3,on
-----+-----+-----
 4,on | 5,on | 6,off
-----+-----+-----
 7,off | 8,on | 9,on
```

Enter position:

You should define a `LightsOff` class, with a main method that works as follows:

- (1) Ask the user for the number of rows and the number of columns in the grid, then create an instance of the `LightsOff` class, with a grid of lighted buttons of the given size, with all of the lights on (implement the grid using a two-dimensional array).
- (2) Display the grid on the screen.
- (3) Repeat until all lights are off:
  - Ask the player to choose which button to press.
  - Read the number of the grid square entered by the player.
  - Update the grid to reflect the result of pressing the chosen button.

Below is a list of methods to implement in your `LightsOff` class, and some questions to think about before you actually start writing the code.

- (1) A constructor that creates a grid of the chosen size. (What should the types of the values stored in the grid be? Should they represent just whether the light is on or off, or should they also include the grid-square numbers?)
- (2) A `printGrid` method that prints the current grid.
- (3) A `choiceOK` method that returns true if the user has selected a legal position.
- (4) A (private) `changeState` method that has two parameters (a row and column number). If that row and column are inside the grid, change the state of that light (from on to off or vice versa). (What single assignment statement can you use to change the state of the light at position  $j$   $k$  of a two-dimensional array named `grid`?)
- (5) A `pressButton` method that simulates pressing the button in the square of the grid selected by the player. (How will this method find the position in the grid that corresponds to the square number selected by the user? Why is it better to have this method call the `changeState` method rather than changing the pressed button and its neighbors directly?)
- (6) A `gameWon` method that returns true if all of the lights in the grid are off.

### 100 point questions

Q: Does `ParentThing.methodA();` compile; if yes, which code segment executes?

A: Does not compile (there is no static `methodA` in the `ParentThing` class).

Q: `parent.methodB("Testing");`

A: Does not compile (there is no `methodB` with a `String` parameter in the `ParentThing` class).

Q: `parent.methodB();`

A: Compiles and executes Code segment 3.

Q: `two.methodB();`

A: Compiles and executes Code segment 5.

Q: `two.methodA();`

A: Compiles and executes Code segment 1.

Q: `kid2.methodA();`

A: Compiles and executes Code segment 9.

Q: `ThingOne.methodB();`

A: Does not compile (there is no static `methodB` in the `ThingOne` class).

Q: `// Replacement for code segment 4  
j = onej;`

A: OK (`j` is a public field of the parent class, `onej` is a field of this class)

Q: `// Replacement for code segment 4  
methodB();`

A: OK (parent class `ParentThing` has a `methodB` with no parameter)

## 200 point questions

Q: `kid1.methodB();`

A: Compiles and executes Code segment 3.

Q: `// Replacement for code segment 4  
methodB( "hello" );`

A: NO (neither ThingOne nor parent class have methodB with a String parameter)

Q: `kid1.methodA(45);`

A: Compiles and executes Code segment 4.

Q: `one.methodA( "5" );`

A: Compiles and executes Code segment 2.

Q: `kid1.methodA( "Smile" );`

A: Compiles and executes Code segment 2

Q: `kid2.methodA( "Hello World!" );`

A: Compiles and executes Code segment 2

Q: `three.methodA(10);`

A: Does not compile (neither ThingThree nor ParentThing has a methodA with an int parameter).

Q: `three.methodB( "String" );`

A: Does not compile (neither ThingThree nor ParentThing has a methodB with a String parameter).

Q: `// Replacement for code segment 9  
k++;`

A: OK (k is a protected field of a superclass)

### 300 point questions

Q: `// Replacement for code segment 4`  
`k = onek;`

A: OK (k is a protected field of the parent class, onek is a field of this class)

Q: `// Replacement for code segment 4`  
`k = onep;`

A: OK (k is a protected field of the parent class, onep is a field of this class)

Q: `// Replacement for code segment 4`  
`p = onep;`

A: NO (p is a *private* field of the parent class)

Q: `// Replacement for code segment 8`  
`j = onej;`

A: OK (j and onej are public fields of superclasses)

Q: `// Replacement for code segment 8`  
`k = onek;`

A: OK (k and onek are protected fields of superclasses)

Q: `// Replacement for code segment 8`  
`onep++;`

A: NO (onep is a *private* field of the parent class)

Q: `// Replacement for code segment 8`  
`methodA(22);`

A: OK (parent class has methodA with an int parameter)

Q: `// Replacement for code segment 9`  
`j = onej;`

A: NO (onej is a field of ThingOne, which is not a superclass)