**CS2030S**

**RECTITATION 1**

1) wrong

a) Class P's changeSelf() method

- abs barrier sits bet client and implementer

- class P is the implementer, Q is the client that makes use of class P

-but Q should not have direct access to class P’s p parameter

- thus the changeAnother method in Q violates the abs barrier

b) client should not have the ability to change the internal information of a class, which in this case the client can use changeSelf() method to edit private int x's value to be set to 1 each time the method is called

-abs barrier is not broken only when obj of type P accesses the instance of variable of another object of same type P

-since both are implementers

-but if 2 objects of different types can access or modify each other’s attributes, then abs barrier is broken

2)

a) LIFO left to right wrong

|  |  |  |
| --- | --- | --- |
| Stack | this = memory\_loc\_of\_v2 | this = memory\_loc\_of\_v1 |
| Heap  Attributes | x =2 | x =1 |
| y = 2 | y = 1 |
|  |  |

-v1 pointer created in stack, v1 object in heap

-v2 pointer created in stack, v2 object in heap

-when the “add” method from v1 is called, a “this” pointer is created in stack which points to v1 (since it is v1’s add method that is called)

-a “v” pointer is created in the stack which points to v2 (since the parameter for the add function is a Vector2D object v).

-operation is done, values in v1 are overwritten

-v pointer (pointing to v2) is destroyed, this pointer (pointing to v1) is destroyed in the stack due to LIFO principle.

b)i)



b) ii)yes

LIFO: left to right

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stack | coord2D[]\_of\_v2 | this = memory\_loc\_of\_v2 | coord2D[]\_of\_v1 | this = memory\_loc\_of\_v1 |
| Heap  Attributes | {2, 2} | coord2D[]\_of\_v2 | {1, 1} | coord2D[]\_of\_v1 |
|  |  |  |  |
|  |  |  |  |

-v1 pointer created in stack, v1 object in heap. v1 object in heap contains a coord2D array {1,1}

-v2 pointer created in stack, v2 object in heap. v2 object in heap contains a coord2D array {3,3}

-when the “add” method from v1 is called, a “this” pointer is created in stack which points to v1 (since it is v1’s add method that is called)

-a “v” pointer is created in the stack which points to v2 (since the parameter for the add function is a Vector2D object v).

-operation is done, values in v1’s coord2D are overwritten to become {3,3}

-v pointer (pointing to v2) is destroyed, this pointer (pointing to v1) is destroyed in the stack due to LIFO principle.

3)

a)



b)

