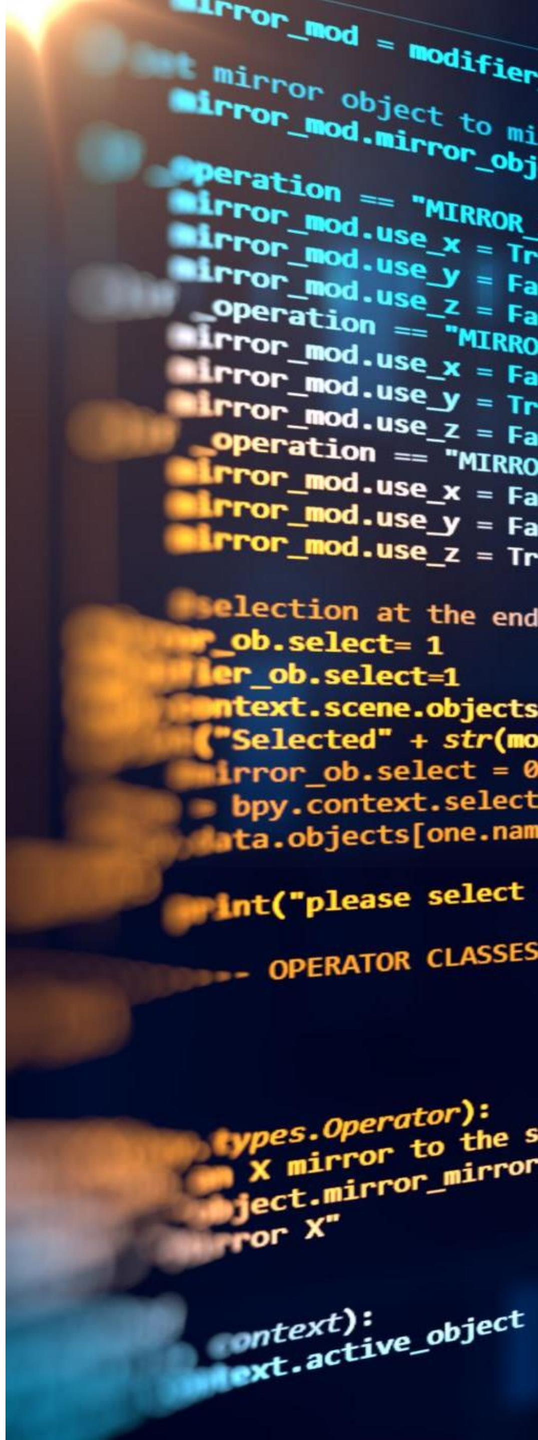
CS 1037
Fundamentals of Computer Science II

Midterm Review

Ahmed Ibrahim



Consider the following Java code:

```
public class AnimalTest {
    public static void main(String[] args) {
        Object animal = new String("Elephant");
        System.out.println(animal.length());
    }
}
```

What will happen when this code is executed?

A) The code will compile and print the string length "Elephant"

B) The code will result in a compilation error

C) The code will throw a runtime NullPointerException

D) The code will throw a runtime ClassCastException

The Object class does not have a length() method, which is why calling animal.length() results in a **compilation error**.

Consider the following Java code:

```
String[] list = new String[3];
list[0] = new String("Hello");
list[1] = "Hello";
list[2] = list[0];

int c = 0;
for (int i = 0; i < 2; ++i) {
    if (list[i] == list[i+1]) c++;   // Compares references
    if (list[i].equals(list[i+1])) c++;  // Compares content
}
System.out.println(c);
```

What will be printed when this code is executed?

A) 1
B) 2
C) 3
D) 4

Consider the following Java code

```
int i = 10;
for (int i = 0; i < 5; i++) {
    System.out.println(i);
}
System.out.println(i);
```

What will happen when the above code is compiled and executed?

A) The code will print numbers from 0 to 4, and then print 10.

B) The code will result in a compilation error due to a variable scope conflict.

C) The code will print numbers from 0 to 5.

D) The code will print 10 five times.

Consider the following Java code:

```
MyObject obj = null;

obj.doSomething();
```

How would you handle the NullPointerException that occurs when this code is executed?

A) Use a try-catch block to catch the NullPointerException.

B) Initialize obj to a new MyObject before calling doSomething().

C) Add a null check before calling doSomething().

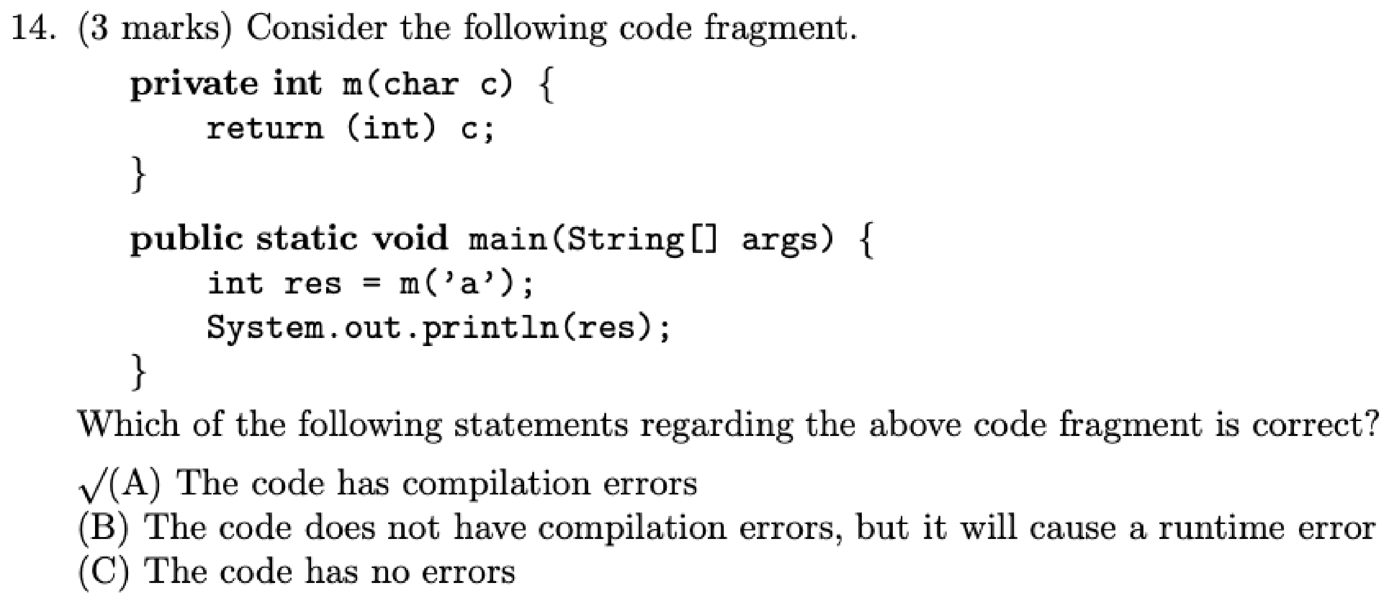
D) All of the above.

```
try {
    obj.doSomething();
} catch (NullPointerException e) {

System.out.println("NullPointerException
caught");
}
```

```
MyObject obj = new MyObject();
obj.doSomething();
```

```
if (obj != null) {
  obj.doSomething();
}
```



14. (3 marks) Consider the following code fragment.

```
private int m(char c) {
    return (int) c;
}
public static void main(String[] args) {
    int res = m('a');
    System.out.println(res);
}
```

Which of the following statements regarding the above code fragment is correct?

[✓ (A) The code has compilation errors]
(B) The code does not have compilation errors, but it will cause a runtime error
(C) The code has no errors

- This question tests your understanding of **access modifiers**, **method accessibility**, and the difference between **instance methods** and **static methods** in Java.

Note:

- The method m(char c) is declared with the **private access modifier**, meaning it can only be accessed within the same class.
- The method m(char c) is an **instance method** because it is not declared static.
- The code also demonstrates **type casting** from char to int. In Java, a char can be cast to an int, where the result is the ASCII or Unicode value of the character.

15. (3 marks) Consider the following code fragment. `Integer` is a wrapper class of Java that represents integer objects. Method `intValue()` of this class returns the `int` value of an `Integer` object. For example, if `Integer intObj = new Integer(3);` the value that `intObj.intValue()` will return is 3.

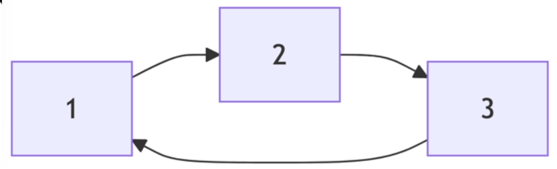
```
Integer[] arr = new Integer[5];
try {
    int s = 0;
    for (int i = 0; i < 5; ++i) s = s + arr[i].intValue();
}
catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Invalid index");
}
catch (NullPointerException e) {System.out.println("Null pointer");}
catch (Exception e) {System.out.println("Exception");}
```

What is printed when the above code is executed?

(A) "Invalid index" [✓ (B) "Null pointer"] (C) "Exception" (D) Nothing
(E) "Invalid index", "Null pointer" and "Exception"

12. (2 marks) Consider the following code fragment. Each node of class **LinearNode** stores an integer value that can be accessed with method **getValue()**.

```
LinearNode<Integer> aNode = new LinearNode<Integer>(1);
LinearNode<Integer> bNode = new LinearNode<Integer>(2);
LinearNode<Integer> cNode = new LinearNode<Integer>(3);
aNode.setNext(bNode);
bNode.setNext(cNode);
cNode.setNext(aNode);
int sum = 0;
LinearNode<Integer> tmp, curr = aNode;
while (curr != null) {
    sum = sum + curr.getValue();
    tmp = curr.getNext();
    curr.setNext(null);
    curr = tmp;
}
```



What is the value of variable **sum** after the above code is executed?

(A) 1 (B) 5 (C) 6 [✓ (D) 7] (E) The program would never terminate

(8 marks) Consider the following code.

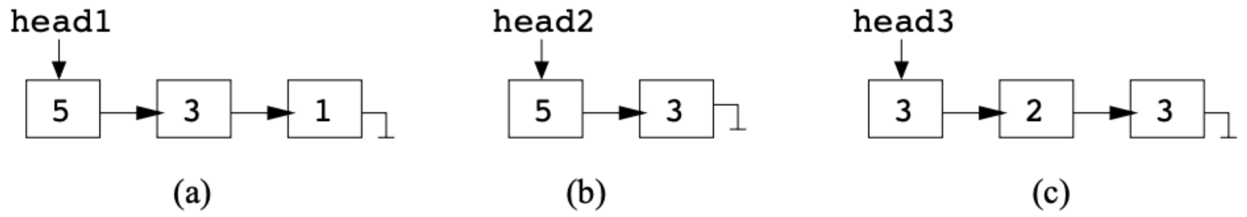
```
public class C {
     private static int i = 0;
     public C() {i = 0;}
     private void method1 (int i) throws Exception1, Exception2 {
         try {
               if (i == 0) throw new Exception1();
               method2 (i);
               this.i = 7;
         } catch (Exception1 e) {
               i = 5;
               System.out.println("Exception 1 caught in method1");
               method2(i);
               if (i == 5) throw new Exception2();
         }
     }
     private void method2 (int x) throws Exception1 {
         int i = 4;
         try {
               if (x > 4) throw new Exception1();
               else throw new Exception2();
         }
         catch (Exception2 e) {System.out.println("Exception 2 caught in method 2");}
     }
     public static void main (String[] args) {
         C varc = new C();
         try {
               varc.method1(0);
               i = 10;
         }
         catch (Exception1 e1) {System.out.println("Exception 1 caught in main");}
         catch (Exception2 e) {System.out.println("Exception 2 caught in main");}
         System.out.println("i = " + i);
    }
}
```

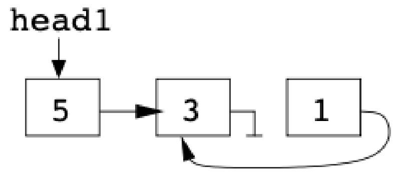
Exception1 and Exception2 are not parent/child classes of each other. Write all the output produced when this program is executed.

20. Consider the following code fragment that modifies a singly linked list in which every node stores an integer value. For a node p, p.getValue() returns the value stored in p.

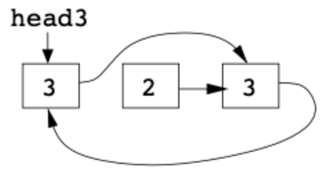
```
public void modify (LinearNode head, int k) {
    LinearNode curr = head, next = curr.getNext();
    while (curr != null)
         if (curr.getValue() == k) {         //Line 1
               curr.setNext(next.getNext()); //Line 2
               next.setNext(curr);           //Line 3
               curr = curr.getNext();        //Line 4
         }
         else {
               curr = next;
               next = curr.getNext();        //Line 5
         }
}
```

Consider the following three linked lists



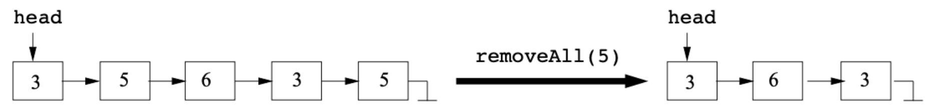


The code throws a null pointer exception when executing Line 2 because next is null.



```
Algorithm removeAll(head,$k$)
    current = head.getNext()
    prev = head
    while current $\neq$ null do
        if current.getValue() = $k$ then {
            prev.setNext(current.getNext())
            current = current.getNext()
        }
        else {
          prev = current
          current = current.getNext()
        }
```

21. (18 marks) Write in Java or in **detailed pseudocode** like the one used in the lecture notes an algorithm `removeAll(head,k)` that receives as parameter a reference `head` to the first node of a singly linked list storing integer values and a value `k` and it removes from the linked list all nodes storing the value `k`. For example, for the singly linked list shown on the left side of the following figure and `k` = 5, the algorithm must remove the nodes storing the value 5, so at the end the list must be as show on the right side of the figure. If no node stores the value `k` then the list must not be modified.

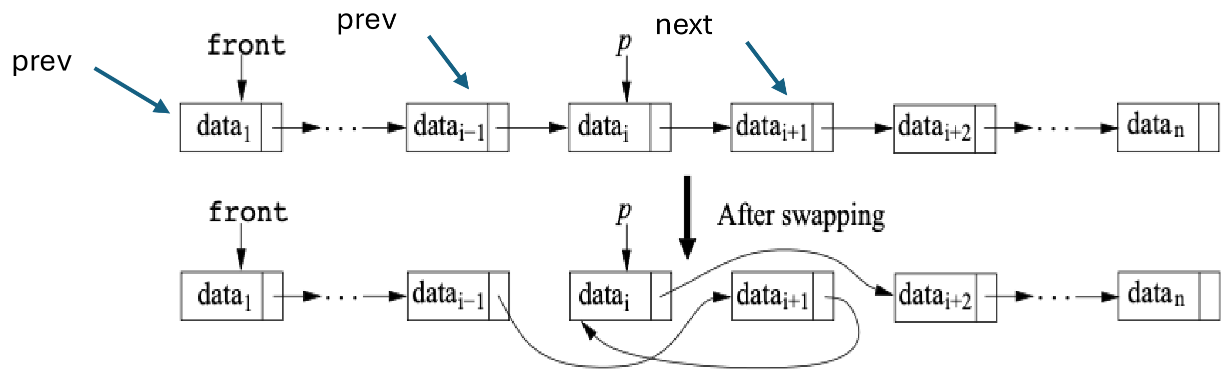


For simplicity, assume that the value `k` is not stored in the first node of the list. Let p and q be references to nodes of the list. **The ONLY methods** that you can use to manipulate the linked list are the following: p.getValue() returns the value stored in p, p.getNext() returns a reference to the next node in the list after p, p.setNext(q) makes node p point to node q.

You CANNOT use any auxiliary data structures (you cannot use an array, stack, queue, another list, and so on). You **CANNOT** create a second linked list and copy the values from the first list to the second one.

Hint. Use two variables: `prev` and `current`. Scan the list and for each node determine whether it stores the value `k` and if so remove it from the list; which pointer needs to change to remove this node? How do you update `prev` and `current`?

23. (18 marks) Given a singly liked list whose first node is referenced by **front**, write an algorithm called **swap**(p) that swaps a given node p with node p.**getNext()**. For example, for the following singly linked list and node p, after swapping nodes p and p.**getNext()** the list should be as in the figure at the bottom.



```
1. Find the node that points to p (we will call this prev).
prev = front

2. while prev.getNext() != p do
// Keep traversing until you find the node before p.
prev = prev.getNext()
end while

3. Store references:
next = p.getNext() // Store the node after p.

4. Perform the swap:
prev.setNext(next) // Set prev to point to next instead of p.
p.setNext(next.getNext()) // Set p's next to next's next node.
next.setNext(p) // Set next's next to p, effectively swapping the two.
```

Thank you