



Technical Test

DEV - E

Date: 26/11/2025 City/State: Olinda - Pernambuc

Course: Computer Science Educational Institution: Cesar School

Course Duration (in years): 4 Current Semester: 6th Graduation Year (expected): 2027

Availability to work: ☐ 20h ☒ 30h ☐ 40h Estimated Start Date: December

Instructions:

This test consists of 8 multiple choice questions, 1 algorithm implementation and 1 non-technical question. The algorithm is worth 60% of the total score. The non-technical question must be answered in Portuguese.

You may use any blank space on this test as a draft.

Use the table below to record your answers.

Good luck!

Answer Sheet

	1	2	3	4	5	6	7	8
A								
B	X							
C				X	X			X
D		X	X			X	X	

Question 1

Given:

```
1. import java.util.Date;
2. public class Example {
3.     public static void main(String args[]) {
4.         Date d1 = new Date (92, 11, 31);
5.         Date d2 = new Date (94, 11, 31);
6.         method(d1, d2);
7.         System.out.println("d1=" + d1.getYear() + "\nd2=" + d2.getYear());
8.     }
9.     public static void method(Date d1, Date d2) {
10.        d2.setYear(98);
11.        d1 = d2;
12.    }
13. }
```

What's the output?

A. d1=92
d2=94

B. d1=92
d2=98

C. d1=98
d2=98

D. d1=98
d2=94

A: Wrong because d2.setYear(98) does change the original d2 object

B: Correct, because in this case, only d2 was changed

C: Wrong because d1 = d2 inside the method does not modify the reference in main. d1 in main remains pointing to the original object (year 92).

D is wrong because D is wrong because d2 becomes 98 in the method, so it cannot be 94.

Question 2

Given:

```
1. //*****
2. // file A.java
3. //*****
4. package a;
5. public class A {
6.     private int x;
7.     protected int y;
8.     public int m1() {return x;}
9. }
10. //*****
11. // file B.java
12. //*****
13. package b;
14. import a.A;
15. public class B extends A {
16.     private int z;
17.     public void m2(A a) {
18.         z = x;
19.         z = y;
20.         z = a.m1();
21.     }
22. }
```

Consider the following statements:

- I. Line 18 is valid because B extends A
- II. Line 19 is not valid because y is protected
- III. Line 20 is valid because m1() is public

- A. Only I and II are correct
- B. Only I and III are correct
- C. Only II and III are correct
- D. Only III is correct <- correct

I. False, because x is private

II. False, y is protected. A protected member is visible in a subclass, even in a different package, when accessed through this/inheritance.

III. True, because m1() is a public method and can be accessed anywhere.

Question 3

What best defines an abstract class?

A is wrong, abstract classes can have method definitions

- A. The class must not have method definitions.
- B. The class must have a method definition returning nothing.
- C. The class must have a constructor that takes no arguments.
- D. The class cannot be instantiated as it is mainly for inheritance.

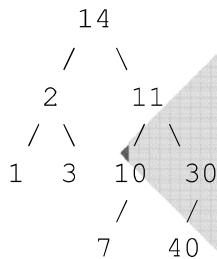
B is wrong too, there is no requirement for methods returning nothing

C is wrong too, abstract classes do not need no-argument constructor

D is correct, an abstract class cannot be instantiated and is intended to be inherited

Question 4

Consider this small binary tree:



What is the order of nodes visited using a post-order traversal?

- A. 1 2 3 7 10 11 14 30 40
- B. 1 2 3 14 7 10 11 40 30
- C. 1 3 2 7 10 40 30 11 14
- D. 14 2 1 3 11 10 7 30 40

C is correct because it follows post-order traversal: visit the entire left subtree (1 3 2), then the entire right subtree (7 10 40 30 11), and finally the root (14).

Question 5

Consider the following statements:

- I. A Binary tree is a tree data structure in which each node has at most two child nodes, usually distinguished as "left" and "right", and a tree with n nodes has exactly $n-1$ branches which means its height is always $n-1$.
- II. A Queue is a FIFO data structure, which means that the first element added to the queue will be the first one to be removed.
- III. A Hash Map is a data structure in which, if there's no collision among the keys, you can always find an element in $O(1)$ time, even in the worst case.

- A. Only I and II are correct
- B. Only I and III are correct
- C. Only II and III are correct
- D. I, II and III are correct

I is incorrect.

A binary tree with n nodes has $n-1$ edges, but its height is not always $n-1$; height depends on shape (balanced vs. skewed).

II is correct.

Queue is FIFO.

III is correct.

If there are no collisions, lookup is $O(1)$ even in the worst case under that assumption.

correct ->

Question 6

In the following code, assume that Queue is not thread-safe, there is more than one Producer thread and more than one Consumer thread running and this program is crashing on runtime. In order to fix the code below how should you fill in lines (1), (2), (3) and (4)?

Global variables	
Queue q; (1)	
Producer thread	Consumer thread
runProducer(){ while(true){ item = new item(); (2) if (q is not full){ q.enqueue(item); (3) } (4) } }	runConsumer(){ while(true){ (2) if (q is not empty){ item = q.dequeue(); (3) } (4) } }

- A. (1) mutex m;
(2) m.lock();
(3)
(4) m.unlock();

Both a mutex and a semaphore can make the queue thread-safe as long as the code locks before the critical section and unlocks after it. Options A and C follow this pattern, ensuring mutual exclusion during enqueue and dequeue operations. Option B does not solve the problem, since adding sleeps does not prevent race conditions. Therefore, the correct answer is D

- B. (1)
(2)
(3)
(4) if(Consumer) sleep(1); else sleep(2);

B: is wrong because just these sleeps won't be solving the problem

- C. (1) semaphore guard;
(2) wait(guard);
(3)
(4) signal(guard);

- D. Alternatives A and C are correct. <- Correct

Question 7

Considering the following tables and data information, what would be the correct result of the SQL command below?

Salesperson			
ID	Name	Age	Salary
1	Abe	61	140,000
2	Bob	34	44,000
5	Chris	34	40,000
7	Dan	41	52,000
8	Ken	57	115,000
11	Joe	38	38,000

Customer			
ID	Name	City	Industry_Type
4	Samsonic	Pleasant	G
6	Panasung	Oaktown	N
7	Samony	Jackson	N
9	Ornange	Hayward	G
8	Hepoul	Cupertino	I

Orders				
Number	Order_Date	cust_id	salesperson_id	Amount
10	8/2/2010	4	2	540
20	5/6/2012	9	7	150
30	3/12/2012	8	5	1,500
40	1/30/2013	4	8	1,800
50	7/14/2009	9	1	460
60	1/29/2012	7	2	2,400
70	2/3/2012	6	7	600
80	4/1/2013	8	2	2,300
90	3/2/2012	6	7	720

```
SELECT Salesperson.Name from Salesperson
WHERE Salesperson.ID NOT IN(
    SELECT Orders.salesperson_id FROM Orders
    INNER JOIN Customer ON Orders.cust_id = Customer.ID
    WHERE Customer.Name = 'Panasung')
AND Salesperson.ID IN
    (SELECT DISTINCT Orders.salesperson_id FROM Orders);
```

A. Bob
Chris
Ken

B. Abe
Bob
Ken

C. Abe
Bob
Chris
Ken
Joe

D. Abe
Bob
Chris
Ken

<- correct

The query returns all salespersons who appear in any order but have never sold to the customer "Panasung." O sold to Panasung, and Joe never appears in any order. Therefore, the result is Abe, Bob, Chris, and Ken, matchir D.

Question 8

Given this output on a Linux terminal:

```
$ cat linux_distributions.txt
Debian distribution
Ubuntu distribution derived from Debian
Fedora distribution
Red Hat Enterprise Linux distribution derived from Fedora
CentOS distribution derived from Fedora
MINIX and Linux operating system
```

What will be the correct result of the command below?

```
$ cat linux_distributions.txt | grep Fedora | sort
```

- A. Fedora distribution
Red Hat Enterprise Linux distribution derived from Fedora
CentOS distribution derived from Fedora
- B. CentOS derived distribution Fedora from
distribution Fedora
derived distribution Enterprise Fedora from Hat Linux Red
- C. CentOS distribution derived from Fedora
Fedora distribution
Red Hat Enterprise Linux distribution derived from Fedora
- D. distribution Fedora
derived distribution Enterprise Fedora from Hat Linux Red
CentOS derived distribution Fedora from

A: Wrong because the lines are not in alphabetical order; "CentOS..." should come before "Fedora..."

B: Wrong because the lines are incomplete and do not match any actual lines from the file.

C: Correct because it shows exactly the three original lines containing "Fedora", in proper alphabetical order.

D: Wrong because the words in the lines are rearranged and no longer match the original file content.

Subsets

Write the function `getSubSets()` to compute and return all subsets of a given set A, which has at most 4 elements.

Use the Set data structure to represent all sets and sub-sets of your solution. A Set is a collection that contains no duplicate elements and the order of elements is irrelevant. Consider the following interface defined for Set:

Method signature	Method description
<code>boolean add(Element e)</code>	Adds the specified element to this set if it is not already present (optional operation).
<code>boolean addAll(Set s)</code>	Adds all elements from s that are not already present in this set.
<code>boolean contains(Element e)</code>	Returns true if this set contains the specified element.
<code>boolean equals(Set s)</code>	Compares the specified set s with this set for equality.
<code>Iterator<Element> iterator()</code>	Returns an iterator over the elements in this set.
<code>boolean remove(Element e)</code>	Removes the specified element from this set if it is present (optional operation).
<code>int size()</code>	Returns the number of elements in this set (its cardinality).
<code>Element[] toArray()</code>	Returns an array containing all of the elements in this set.

Table: Set interface

Input example:

`A=[1,2,3]`

Output for the given example:

`[[1,2,3], [1,2], [1,3], [2,3], [1], [2], [3], []]*`

* this is the content of the Set which should be returned by the function.

Your proposed solution can be written in pseudo-code or any well-known language (C, C++, Java, etc) and you are free to implement any auxiliary functions. Besides, write down a comment to the main function explaining how your function will work like the one below.

```
/**
 * The function below will ...
 * - Obtain the input
 * - Iterate over the elements
 * ...
 * - Print the output and return ...
 */
```

Algorithm Solution

```
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;

public class SubsetsExample {

    public static Set<Set<Element>> getSubSets(Set<Element> inputSet) {
        Set<Set<Element>> allSubsets = new HashSet<>();
```

```
        // Convert set to array to access elements by index
        Element[] elements = inputSet.toArray(new Element[0]);
        int n = elements.length;
        int totalMasks = 1 << n; // 2^n subsets
```

```
        // For each bitmask, build the corresponding subset
        for (int mask = 0; mask < totalMasks; mask++) {
            Set<Element> subset = new HashSet<>();
```

```
            for (int i = 0; i < n; i++) {
                if ((mask & (1 << i)) != 0) {
                    subset.add(elements[i]);
                }
            }
        }
```

```
        allSubsets.add(subset);
    }
```

```
    return allSubsets;
}
```

```
/**
```

```
 * The main method will:
```

```
 * 1. Build the input set A.
```

```
 * 2. Call getSubSets(A) to compute all subsets of A.
```

```
 * 3. Iterate over the returned set of subsets and print each subset.
```

```
*/
```

```
public static void main(String[] args) {
    Set<Element> a = new HashSet<>();
    a.add(new Element(1));
    a.add(new Element(2));
    a.add(new Element(3));
```

```
    Set<Set<Element>> subsets = getSubSets(a);
```

```
    for (Set<Element> subset : subsets) {
        System.out.println(subset);
    }
}
```

```
// Example Element implementation just for completeness
```

```
class Element {
    private final int value;

    public Element(int value) {
        this.value = value;
    }

    @Override
    public String toString() {
        return Integer.toString(value);
    }

    @Override
    public int hashCode() {
        return Integer.hashCode(value);
    }

    @Override
    public boolean equals(Object obj) {
        if (!(obj instanceof Element)) {
            return false;
        }
        Element other = (Element) obj;
        return this.value == other.value;
    }
}
```


Qual a disciplina que você mais gostou de cursar na faculdade e por quê? (Responder em português)

No segundo semestre, cursei a disciplina de Fundamentos de Software, que abrangia todo o ciclo de desenvolvimento de um produto, do planejamento inicial até a entrega. Começamos pelo básico, como escrever histórias de usuário e definir requisitos, e avançamos para etapas mais complexas, como implementação, testes, sprints, reuniões de alinhamento e, claro, muitos desafios com Django, o framework que estávamos aprendendo a usar.

O que tornou essa disciplina especialmente marcante foi a combinação de teoria e prática. Tivemos contato com métodos ágeis, pipelines de desenvolvimento, estratégias de teste, versionamento e até processos de deploy, sempre trabalhando em equipe para lidar com problemas reais. Cada etapa exigia adaptação, aprendizado rápido e bastante colaboração.

No fim, considero essa cadeira até o momento a mais enriquecedora da faculdade. Foi desafiadora e ao mesmo tempo muito satisfatória. Aprendi não só sobre tecnologia, mas também sobre comunicação, organização e trabalho em equipe.

