**Preguntas Teóricas de Taller:**

**Imperativo:**

**Ordenación de vectores  
1. ¿En qué consiste la ordenación de vectores?**

Siguiendo un patrón único intercambio los elementos de un vector ya cargado para acomodarlos con un cierto criterio.

El proceso por el cual, un grupo de elementos puede ser ordenado se conoce como algoritmo de ordenación.

*Pseudocodigo:*

1. Ordenación de menor a mayor.

**Ordenar** (v: tipo\_vector; n: valor\_de\_dimL);

Repetir desde i: =2 hasta n

guardar dato (a ordenar)

j: = i-1

Mientras (j > 0) y (v[j] > dato a ordenar)

v[j+1]: = v[j]

j: = j - 1

guardar dato en v[j+1]

**2. ¿Qué es y qué hace un algoritmo de ordenación?**

Serie de pasos los cuales consiguen dado un vector de elementos, intercambiar esos elementos en distintas posiciones tal que el resultado será el mismo vector con sus elementos ordenados en base a un criterio.

**3. ¿Mencione al menos dos algoritmos de ordenación de vectores? ¿Cómo funcionan? Ejemplifique**

Selección, intercambio, inserción.

Selección: en cada búsqueda elige el menor elemento del vector y lo sitúa en su lugar correspondiente.

Intercambio: Se comparan e intercambian datos adyacentes según corresponda. Requiere n-1 pasadas sobre el vector. Si se quiere ordenar el vector de menor a mayor, al final de cada pasada el mayor queda al final del arreglo (no es necesario volver a compararlo).

Inserción: Se parte de una secuencia de dos ítems y se ordena. En cada pasada se “agrega” un ítem y se inserta en la posición correspondiente en el arreglo ordenado.

**Recursión**

**1. Defina el concepto de Recursión. Explique las principales características que se presentan en un algoritmo recursivo.**

La recursión es una metodología para resolver problemas. Permite resolver un problema P por resolución de instancias más pequeñas P1, P2, …Pn del mismo problema.

El problema Pi (subproblema) es de la misma naturaleza que el problema original, pero en algún sentido es más simple.

Una solución recursiva resuelve un problema por resolución de instancias más pequeñas del mismo problema.

Un algoritmo recursivo involucra:

-Un caso base.

-Al menos una condición de terminación (implícita / explícita)

-Al menos una auto invocación (llamada recursiva). Se debe garantizar que en un número finito de auto-invocaciones se alcanza la condición de terminación.

**2. ¿Una solución recursiva puede tener más de un caso base? Justifique.**

SI. Haber llegado al final, que el elemento no exista o el vector este vacío y cuando encuentro el dato a buscar.

**3. Describa dos ejemplos de problemas recursivos con más de un caso base. Implemente dicha solución.**

Búsqueda de un elemento en un árbol, vector, lista (Que no exista el elemento, encontrarlo) y una función factorial (que el exponente sea 1, o que sea 0 son 2 casos base). Ejemplo de implementación de

Búsqueda de un elemento en un árbol:

Procedure Buscar (a: árbol; var p: árbol; dato: integer);

Begin

If (a=nil) then

p:= nil (Arbol Vacio)

else

if (dato= a^.dato) then

p:= a (Dato encontrado)

else

if (dato < a^.dato) then

Buscar (a^.HI,p,dato)

else

Buscar(a ^.HD,p,dato);

End;

**4. Enuncie un problema que justifique una solución recursiva e impleméntela.**

Un problema en el cual se puede resolver mediante la recursión es el factorial de un número. Su implementación es:

Function potencia (nro,e: integer): real;

begin

if (e = 0) then

potencia:= 1

else

if (e = 1)then

potencia:=numero

else

potencia := nro \* potencia(nro,e-1);

end;

**Merge de listas**

**1. ¿En qué consiste el Merge? Mencione cuales son las precondiciones que se requieren para realizar la operación.**

La operación de Merge consiste en generar una nueva estructura de datos (arreglos, listas) ordenada a partir de la mezcla de dos o más estructuras de datos previamente ordenadas.

Las estructuras que se combinan guardan el mismo orden lógico interno (por ejemplo, datos ordenados alfabéticamente).

La precondición para realizar la operación es que las estructuras a mezclar estén ordenadas por el mismo criterio de ordenación.

Otro tipo de Merge es el Acumulador, que la única diferencia que tiene con un Merge es que **primero acumula cantidad de un mismo elemento** antes de guardarlo en la nueva estructura.

**2. Explique detalladamente los pasos a seguir para realizar la operación de Merge de dos listas simplemente enlazadas.**

Debemos hallar el mínimo elemento entre las dos listas, se inserta el elemento en la nueva estructura y se elimina de la anterior (elimino una vez que se cual es definitivamente el menor).

Se repite él algoritmo hasta que ambas listas no tengan elementos, es decir, que las dos listas estén en NIL.

**3. ¿En qué se diferencian las operaciones de Merge y Merge acumulador? Ejemplifique.**

La operación de Merge consiste en agrupar en una misma estructura elementos provenientes de distintas, ordenados por un criterio. Un Merge acumulador consiste en acumular elementos con un mismo criterio e insertarlos en una nueva estructura siguiendo una condición de orden.

**Árbol binario de búsqueda**

**1. ¿Qué es un árbol?**

Un árbol es una estructura de datos cuyas características son:

**1.homogénea** (todos los elementos son del mismo tipo)

**2.dinámica** (puede aumentar o disminuir su tamaño durante la ejecución del programa)

**3.no lineal** (cada elementos puede tener 0,1 o más sucesores )

**4.acceso secuencial**

que satisface tres propiedades:

Cada elemento del árbol se relaciona con cero o más elementos (hijos).

Si el árbol no está vacío, hay un único elemento (raíz) y que no tiene padre (predecesor).

Todo otro elemento del árbol posee un único padre y es un descendiente de la raíz.

**2. ¿Qué es un árbol binario?**

Un árbol binario es una estructura de datos en la cual cada nodo puede tener a lo sumo 2 hijos (un hijo izquierdo y/o un hijo derecho). No pueden tener más de dos hijos (de ahí el nombre "binario").

**3. ¿Qué es un árbol binario de búsqueda?**

Un ABB es un árbol binario el cual sus elementos se encuentran ordenados.

Cada nodo tiene un valor que:

* Es más grande que el valor de todos los nodos del subárbol izquierdo
* Es menor que el valor de todos los nodos del subárbol derecho

Utilidad más importante ➔ Búsquedas el tiempo medio es O (log n)), en el peor de los casos puede ser de O(n) por ejemplo en caso de que se inserten valores que ya están ordenados

**5. Describa detalladamente la operación Insertar un elemento en un árbol binario de búsqueda.**

Primero necesito dos parámetros, uno sería el árbol pasado por referencia y el otro parámetro pasado por valor es el dato que quiero insertar. Utilizo una variable local de tipo árbol para guardarme los datos recibidos por parámetro y asegurarme de no perder datos. Luego pregunto si el árbol recibido como parámetro está vacío, en que caso de ser cierto, guardo los datos en la variable local creada, pongo en NIL los hijos, izquierdo y derecho de esa variable local creada y luego le asigno esa variable local con los datos ya guardados a la variable que recibí como parámetro. En caso de que el árbol que recibí como parámetro no este vacío, invoco la recursión preguntando si el dato que recibí es mayor o menor al dato del árbol para poder ver donde lo ubico, es decir, si el dato que recibo va para el hijo izquierdo o derecho.

**Árbol binario de búsqueda**

**1. ¿Cuáles son los recorridos clásicos sobre un ABB?**

Los distintos recorridos permiten desplazarse a través de todos los nodos del árbol de tal forma que cada nodo sea visitado una y solo una vez.

Existen varios métodos que se diferencian en el orden que se visitan:

En orden (hijo izquierdo, raíz, hijo derecho),

pre orden (raíz, hijo izquierdo, hijo derecho)

post orden (hijo izquierdo, hijo derecho, raíz).

**2. Describa detalladamente la operación de Imprimir en Orden en un árbol binario de búsqueda.**

Para imprimir en orden en un árbol de búsqueda, lo puedo utilizar es un método recursivo que recibe el árbol como parámetro, pero por valor, y a partir de ahí, se auto invoque con el HI hasta llegar al final, ósea un subárbol vacío, y a partir de ahí imprima los valores subiendo hacia la raíz, y luego recursivamente, realice lo mismo con la pata o subárbol derecho.

**3. Plantee un ejemplo de árbol binario de búsqueda y explique detalladamente cuales serían los pasos a seguir para realizar un recorrido acotado en dicho árbol.**

Para realizar un recorrido acotado, debo tener en cuenta tres recursiones, que este entre la raíz, que este a la derecha, o a la izquierda. También que no sea nil.

Hemos analizado algunas situaciones que obligan a recorrer todos los nodos del árbol, por ejemplo, imprimir los nodos del árbol. Cuando necesitamos mostrar los datos que están comprendidos entre dos valores determinados del árbol, debemos hacer lo siguiente:

recorridoAcotado (árbol, inf, sup);

**si** árbol no está vacío

**si** el valor en árbol es >= inf

**si** el valor en árbol es <= sup

mostrar valor

recorridoAcotado (hijo\_izq\_arbol, inf, sup);

recorridoAcotado (hijo\_der\_arbol, inf, sup);

**sino**

recorridoAcotado (hijo\_izq\_arbol, inf, sup);

**sino**

**recorridoAcotado (hijo\_der\_arbol, inf, sup);**

**4. Suponga que dispone de un árbol binario con datos de alumnos (Apellido, Nombre y DNI) ordenado por DNI. Detalle los pasos necesarios para conocer el apellido y nombre del alumno que posee el mínimo valor de DNI.**

Recorro el árbol mientras no sea nil, hacia el hijo izquierdo.

**Árbol binario de búsqueda**

**1. Detalle los pasos necesarios para borrar un elemento de un árbol binario de búsqueda.**

Cuando es hoja, cuando tiene un hijo, cuándo tiene los dos.

Cuando el elemento es una hoja, la elimino. Cuando tiene dos hijos busco el más grande del sub árbol izquierdo del elemento a borrar, lo reemplazo y lo elimino. Cuando tiene un hijo lo reemplazo.

**A la hora de borrar un elemento del árbol binario hay que considerar las siguientes situaciones:**

1. Si el nodo es una **Hoja:**

Se puede borrar inmediatamente (actualizando direcciones)

1. Si el nodo tiene un **Hijo**:

Si **el** **nodo tiene un hijo**, el nodo puede ser borrado después que su padre actualice el puntero al hijo del nodo que se quiere borrar.

1. Si el nodo tiene dos Hijos:
2. Se busca el valor a borrar.
3. Se busca y selecciona el hijo más a la izquierda del subárbol derecho del nodo a borrar (o el hijo más a la derecha del subárbol izquierdo).
4. Se intercambia el valor del nodo encontrado por el que se quiere borrar.
5. Se llama al borrar a partir del hijo derecho con el valor del nodo encontrado.

borrarElemento (arbol, dato, resultado)

**Si** arbol es vacío

No se encontró el dato a buscar

**Sino**

**Si** el dato en arbol es > dato

borrarElemento (hijo\_izq\_del\_arbol, dato, resultado)

**Sino**

**Si** el dato en arbol es < dato

borrarElemento (hijo\_der\_del\_arbol, dato, resultado)

**Sino**

{se encontró el dato a borrar}

**Si** tiene solo hijo derecho…

**Sino**

**Si** tiene solo hijo izquierdo…

**Sino**

Buscar el mínimo del subárbol derecho

Reemplazar el valor en arbol por el mínimo

**borrarElemento (hizo\_der\_del\_arbol, mínimo, resultado);**

**Programación orientada a objetos**

**1. ¿Qué es un objeto? ¿Cómo se compone? Ejemplifique.**

*Objeto:* abstracciónde un objeto del mundo real, definiendo qué lo caracteriza (estado interno) y qué acciones sabe realizar (*comportamiento*). Es una entidad que combina en una unidad.

*Estado interno*: compuesto por datos/atributos que caracterizan al objeto y relaciones con otros objetos con los cuales colabora. Se implementan a través de variables de instancia.

*Comportamiento:* acciones o servicios a los que sabe responder el objeto. Se implementan a través de métodos de instancia que operan sobre el estado interno. Los servicios que ofrece al exterior constituyen la *interfaz.*

Representación abstracta de cualquier cosa que nos rodea. Cualquier cosa que podamos nombrar y que nos sea de utilidad. Todo es un objeto.

- Características físicas, nombre, estado, atributos, comportamientos.

***Encapsulamiento (ocultamiento de información)***: Se oculta la implementación del objeto hacia el exterior. Desde el exterior sólo se conoce la interfaz del objeto. Facilita el mantenimiento y evolución del sistema ya que no hay dependencias entre las partes del mismo.

**2. ¿Qué es una clase? ¿Qué es una instancia?**

- Una clase es la implementación de las características, el estado y comportamiento de un objeto.

- La instancia es la representación viva de ese objeto. Quien almacena valores y se ejecuta es la instancia de la clase.

Una *clase* describe un conjunto de objetos comunes (mismo tipo). Consta de:

* La declaración de las variables de instancia que implementan el estado del objeto.
* La codificación de los métodos que implementan su comportamiento.

Un objeto se crea a partir de una clase (el objeto es *instancia* de una clase).

**3. ¿Cómo se crea un objeto? ¿Cuáles son los pasos que se siguen en Java para la creación de un Objeto?**

Utilizando el new, Alumno a = new Alumno (…);

* Declarar variable para mantener la referencia:

NombreDeClase miVariable;

* Enviar a la clase el mensaje de creación y guardar referencia:

miVariable= new NombreDeClase(valores para inicialización);

* Se puede unir los dos pasos anteriores:

NombreDeClase miVariable= new NombreDeClase(…);

La *instanciación* se realiza enviando un mensaje de creación a la clase.

* Reserva de espacio para el objeto.
* Ejecución el código inicializador o ***constructor****.*

Un ***constructor*** puede tomar valores pasados en el mensaje de creación. Inicializa el objeto (variables de instancias) con valores recibidos.

Devuelve la referencia al objeto.

**Asociar la referencia a una variable (a través de ella podemos enviarle mensajes al  
objeto).**

**Secuencia de pasos en la instanciación (creación de objeto):**

* *Reserva de Memoria.* Las variables de instancia se inicializan a valores por defecto o explícito (si hubiese).
* *Ejecución del Constructor* (código para inicializar variables de instancia con los valores que enviamos en el mensaje de creación).
* ***Asignación de la referencia a la variable.***

**Constructores**

**1. Describa el concepto de constructor (objetivo, sintaxis de declaración, sintaxis de uso, concepto de sobrecarga).**

Es un método especial porque no se llama, sino que es invocado por el operador new al momento de crear una instancia. Su objetivo es darle un valor inicial a la instancia. En su sintaxis tiene el mismo nombre que la clase y no tiene tipo de retorno. Se diferencia por la firma.

-Todo constructor debería tener el mismo nombre que el de la clase, puede haber más de uno. No tiene un tipo de retorno. Respetar las firmas.

Firma: nombre del método y lista de parámetros.

Java diferencia los métodos sobrecargados con base en el número y tipo de parámetros o argumentos que tiene el método y no por el tipo que devuelve

**2. ¿Es posible generar una clase sin un constructor? Justifique.**

Sí es posible, debido a que el compilador genera un constructor por defecto que inicializa las variables en valores por defecto.

**Herencia**

**1. Defina el concepto de herencia. Describa dos ejemplos donde la herencia sea de utilidad para reutilizar código.**

Es un mecanismo mediante el cual una clase adopta no solo su propio estado interno sino además el comportamiento de otra clase, la cual se la denomina clase base, superclase o clase padre. A la clase que hereda los métodos y atributos se la denomina clase derivada o subclase.

Una Subclase puede reeimplementar los métodos definidos por su superclase. Se heredan tanto métodos como atributos.

Este mecanismo posibilita la reutilización de código y hace que los programa sea más fácil de mantener, de decir, si hay un error o se quiere emplear una modificación, se corrige directamente en la clase base.

La herencia también permite definir comportamiento obligatorio a sus clases derivadas.

* La herencia Permite que la clase *herede* características y comportamiento (atributos y métodos) de otra clase (clase padre o superclase). A su vez, la clase define características y comportamiento propio.

**2. Describa los conceptos de clase abstracta y método abstracto, indicando además para qué se usan y sintaxis de declaración en Java.**

**Clases Abstractas:** Es una clase que no puede ser instanciada (no se pueden crear objetos). Define características y comportamiento común para un conjunto de clases (subclases). Puede definir ***métodos abstractos*** (sin implementación) que *deben* ser implementados por las subclases.

La clase abstracta es una clase que no puede ser instanciada (no se pueden crear objetos/instancias).

Uso: define características y comportamiento común para un conjunto de clases

(subclases).

**Método abstracto: dice que hace un objeto no dice como hacerlo.**

public abstract class NombreClase {}

public abstract tipoRetorno nombreMetodos ()

**3. Describa una jerarquía de clases donde aplique el uso de clase abstracta y método abstracto. Justifique la aplicación.**

Una clase abstracta figura con sus subclases figuras concretas, puede definir variables de instancia como color de relleno, un campo común en todas la clases derivadas y un método abstracto área donde cada clase derivaba sabe responder a este mensaje , por ejemplo con cuadrado b\*h, un círculo pi\*r², triangulo b\*h/2,etc..

* ***Encapsulamiento:*** Permite construir componentes autónomos de software, es decir, independientes de los demás componentes. La independencia se logra ocultando detalles internos (implementación) de cada componente. Una vez encapsulado, el componente se puede ver como una caja negra de la cual sólo se conoce su interfaz.
* ***Herencia:*** Permite definir una nueva clase en términos de una clase existentes. La nueva clase hereda automáticamente todos los atributos y métodos de la clase existente, y a su vez puede definir atributos y métodos propios.
* ***Polimorfismo:*** Objetos de clase distintas puede responder a mensajes con selector (nombre) sintácticamente idéntico de distinta forma. Permite realizar códigos genéricos, altamente reusables.
* ***Binding Dinámico:*** Mecanismo por el cual se determina en tiempo de ejecución el método (código) a ejecutar para responder a un mensaje.

***Beneficios de la Programación Orientada a Objetos (POO):***

* **Natural:** El programa queda expresado usando términos del problema a resolver, haciendo que sea más fácil de comprender.
* **Fiable:** La POO facilita la etapa de prueba del SW. Cada clase se puede probar y validar independientemente.
* **Reusable:** Las clases implementadas pueden reusarse en distintos programas. Además, gracias a la herencia podemos reutilizar el código de una clase para generar una nueva clase. El polimorfismo también ayuda a crear código más genérico.
* **Fácil de mantener: Para corregir un problema, nos limitamos a corregirlo en un único lugar.**

**Programación concurrente**

**Concurrencia:** Es la característica de los sistemas que indica que múltiples procesos/tareas

pueden ser ejecutados al mismo tiempo y pueden cooperar y coordinarse para

cumplir la función del sistema.

**1. Enumere y describa tres características que generalmente existen en un programa concurrente.**

Un programa concurrente se basa en la simultaneidad en la ejecución de múltiples procesos.

Sus características son:

1.la comunicación entre procesos,

2.la sincronización y

3.debe haber más de un proceso (obligatorio).

El programa se divide en tareas (2 o más), las cuales se ejecutan al mismo tiempo y realizan acciones para cumplir un objetivo común. Para esto pueden: compartir recursos, coordinarse y cooperar

**2. Para realizar la comunicación entre procesos ¿qué técnicas existen? Explíquelas.**

Para la comunicación entre procesos hay dos técnicas:

**memoria compartida** (donde hay una variable en común, donde un proceso tiene que escribir y otro tiene que leer), la comunicación se realiza a través de la memoria (En r-info seria las esquinas)

**memoria distribuida** (se basa en la comunicación entre procesos mediante mensajes donde un proceso lo envía y otro lo recibe). (los procesos se comunican a través de mensajes explícitos)

**3. Si tuviera que compartir información entre dos robots del ambiente dado en el curso. ¿Qué opciones tendría? Detalle su respuesta.**

Tendría las opciones de memoria compartida (comunicación a través de las esquinas, se necesita sincronismo para asegurar el intercambio de datos)

y memoria distribuida (comunicación a través de mensajes explícitos EnviarMensaje, RecibirMensaje).

**4. Suponga que en el entorno R-info (el que utilizó en la  
práctica) no existiera el pasaje de mensajes, usted ¿podría afirmar que los procesos no se pueden comunicar? JUSTIFIQUE su respuesta.**

Los procesos sin el envío de mensajes podrían comunicarse igual, ya que en r-Info las esquinas son los recursos compartidos, y mediante el depósito de una flor o papel el otro robot debería hacer una u otra cosa.

**Programación concurrente**

**1 Explique detalladamente cómo funciona la recepción de un mensaje (sintaxis, variantes).**

**Sintaxis:** RecibirMensaje (datoQueMeEnvían, quienMeLoEnvía),

1.si un proceso hasta no recibir el mensaje no puede seguir su curso, se denomina a la recepción de mensajes como un método SINCRONICO

2.si un proceso puede seguir ejecutándose por más que no haya recibido un mensaje, la recepción se considera ASINCRONICA-

Si la instrucción traba el robot hasta que el robot receptor reciba el mensaje que espera, esto se denomina método Sincrónico. Por otro lado, el método asincrónico el robot sigue su ejecución a pesar de no recibir ningún mensaje.

R-Info utiliza Recepción de mensajes sincrónicos.

**2. ¿Cuántos tipos de áreas existen en el ambiente r-info? ¿Para qué se usa cada una? ¿Qué consideraciones se debe tener en cada una de ellas?**

Existes tres tipos de áreas:

***Privada*** (solo puede acceder un robot a su área privada) hay que considerar que solo un robot la acceda,

***Compartida*** (todos los robots tienen acceso),

***Parcialmente Compartida*** (solamente la pueden acceder un subconjunto de robots) tengo que considerar bloquear esquinas para evitar chocarse con otros robots (semi área Compartida)

**3. ¿Qué componente de hardware de las computadoras actuales se ve relacionado con los tipos de áreas que se pueden definir en el entorno R-info? Justifique.**

Los tipos de áreas que hay en el entorno se pueden relacionar con la memoria.

La cache se puede relacionar con un área **parcialmente compartida** (puede ser compartida por varios procesadores)

la memoria principal hace referencia a un área **Compartida**.

Los registros de los procesadores hacen referencia a un área **Privada**.

**Programación concurrente**

**1. Describa un problema en el cual se justifique utilizar un área privada y un área parcialmente compartida.**

Una carrera de robots donde cada recorrido es un área privada y después deben depositarlo en una esquina, sumando la existencia de un robot jefe que no accede a la esquina

**2. Escriba un enunciado en el cual se justifique utilizar un área exclusiva y un área compartida.**

Mismo ejemplo que el anterior, pero sin el jefe.

**3. Escriba el enunciado de un problema en el cual se justifique utilizar áreas parcialmente compartidas.**

Dos áreas compartidas donde cada área tiene dos robots

**4. Escriba el enunciado de un problema en el cual se justifique utilizar SOLO áreas compartidas.**

Todos los robots manejándose en la misma ciudad

**Programación concurrente**

**1. Describa el mecanismo usado en r-info para que dos o más robots tomen objetos de una esquina para depositarlos en otra.**

Bloquear – Entrar – Hacer – Salir – Desbloquear (Siempre es así el orden)

**2. Describa el mecanismo usado en r-info para que un robot actúe como coordinador de dos o más robots (envío y recepción de datos**)

Para que un robot actúe como coordinador de dos o más robots primero debe enviar un mensaje a cada robot un identificador (id).