UNIVERSIDAD NACIONAL DE LA MATANZA
DEPARTAMENTO DE INGENIERÍA E INVESTIGACIONES TECNOLÓGICAS

GUÍA DE TRABAJOS PRÁCTICOS

Carrera: Ingeniería Informática

Materia: PARADIGMAS DE PROGRAMACIÓN-3646

Año lectivo: 2023

Docentes:

Aubin, Verónica Inés
Videla, Lucas
Gasior, Federico
Lanzillotta, Hernán

Jefe de Cátedra:

Aubin, Verónica Inés

Carrera **INGENIERIA EN INFORMATICA**

Asignatura 3646 - Paradigmas de Programación

Tema Guía Diagnóstico

Unidad 2

Objetivo Evaluar el buen uso de arrays multidimensionales, y sus formas de recorrerlos.

Competencia/s a desarrollar

Genéricas

- **Competencias tecnológicas**

1. Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.
2. Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información / informática.
3. Gestión, planificación, ejecución y control de proyectos de ingeniería en sistemas de información / informática.
4. Utilización de técnicas y herramientas de aplicación en la ingeniería en sistemas de información / informática.

- **Competencias sociales, políticas y actitudinales**

5. Desempeño en equipos de trabajo.
6. Comunicación efectiva.
7. Actuación profesional ética y responsable.
8. Aprendizaje continuo.
9. Desarrollo de una actitud profesional emprendedora.

Descripción de la Actividad

Tiempo estimado de resolución: 2 horas

Metodología:
Resolución de problemas de programación con una guía autoexplicativa.

Forma de entrega:
Entrega presencial opcional durante el período de clase destinado a tal fin.

Metodología de Corrección y Retroalimentación:
Se realizará una puesta en común de ejercicios seleccionados. Se utilizará el código provisto por estudiantes voluntarios para analizar distintas posibles resoluciones. Se utilizarán técnicas de pair programming y mob programming para resolver los ejercicios que se seleccionen para hacer en la clase.

Bibliografía

-

Guía de Ejercicios: Diagnóstico

https://github.com/paradigmas-de-programacion/guia-diagnostico

Resolver los siguientes ejercicios utilizando el lenguaje de programación Java, con el paradigma imperativo. Encontrarás una estructura de proyecto conveniente para ese objetivo.

Ejercicio 01

Escribir un método en Java que permita comprobar si la diagonal principal de una matriz cuadrada de enteros tiene en cada posición un valor igual a la suma de todos los valores de las posiciones anteriores de dicha diagonal.

Por ejemplo, la siguiente matriz comprueba la regla:

1 2 3

4 1 6

7 8 2

Pero esta otra, no la comprueba:

1 2 3

4 2 6

7 8 2

Ejercicio 02

Escribir un método en Java que permita comprobar si la multiplicación de todos los valores de una matriz numérica, dará 0 o no.

Ejercicio 03

Escribir un método en Java que de una matriz numérica dada, devuelva una matriz con la misma cantidad de elementos, pero donde cada valor es la suma de sus adyacentes originales (arriba, abajo, izquierda, y derecha; si existen)

Ejemplo: Para la matriz

8 2 -3 4

5 -6 -6 20

21 1 -5 0

La salida debe ser

15 1 -3 21

28 -4 0 18

27 11 -10 15

Ejercicio 04

Escribir un método en Java que de una matriz numérica dada, devuelva un vector con n elementos, donde cada elemento es la moda de una fila. Si hubiese más de una moda, se deberá utilizar la de mayor valor

Ejemplo: Para la matriz

1 2 3 4

5 -6 -6 20

1 1 10 10

La salida debe ser

4 -6 10

Ejercicio 05

Escribir un método en Java que de una matriz cuadrada dada de dimensión nxn, devuelva una matriz con 2n-1 filas, donde cada una tendrá los datos de una de las diagonales (de abajo a la izquierda hacia arriba a la derecha)

Ejemplo: Para la matriz

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

La salida debe ser

13

9 14

5 10 15

1 6 11 16

2 7 12

3 8

4

Carrera **INGENIERIA EN INFORMATICA**

Asignatura 3646 - Paradigmas de Programación

Tema Guía de Objetos

Unidad 3

Objetivo Experimentar el buen uso del encapsulamiento y conceptos básicos del paradigma.

Competencia/s a desarrollar Genéricas

- **Competencias tecnológicas**

1. Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.
2. Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información / informática.
3. Gestión, planificación, ejecución y control de proyectos de ingeniería en sistemas de información / informática.
4. Utilización de técnicas y herramientas de aplicación en la ingeniería en sistemas de información / informática.

- **Competencias sociales, políticas y actitudinales**

5. Desempeño en equipos de trabajo.
6. Comunicación efectiva.
7. Actuación profesional ética y responsable.
8. Aprendizaje continuo.
9. Desarrollo de una actitud profesional emprendedora.

Descripción de la Actividad

Tiempo estimado de resolución: 2 horas

Metodología:
Resolución de problemas de programación con una guía autoexplicativa.

Forma de entrega:
Entrega presencial opcional durante el período de clase destinado a tal fin.

Metodología de Corrección y Retroalimentación:
Se realizará una puesta en común de ejercicios seleccionados. Se utilizará el código provisto por estudiantes voluntarios para analizar distintas posibles resoluciones. Se utilizarán técnicas de pair programming y mob programming para resolver los ejercicios que se seleccionen para hacer en la clase.

Guía de Ejercicios: Objetos

https://github.com/paradigmas-de-programacion/guia-objetos

Resolver los siguientes ejercicios utilizando el lenguaje de programación Java, con el paradigma imperativo. Encontrarás una estructura de proyecto conveniente para ese objetivo.

Ejercicio 01

Implementar la clase Nota para cumplir con la interfaz proporcionada.

Agregar a la clase Nota el método:

```
/**
 * pre : nuevoValor está comprendido entre 0 y 10.
 * post: modifica el valor numérico de la Nota, cambiándolo
 *       por nuevoValor, siempre y cuando nuevoValor sea superior al
 *       valor numérico actual de la Nota.
 */
public void recuperar(int nuevoValor) { }
```

Realizar todas las pruebas que considere convenientes. Se pueden agregar métodos privados.

Ejercicio 02

Implementar la clase Punto. Un Punto en el plano posee coordenada X y coordenada Y. Proporcionar métodos para:

1. obtener y cambiar cada una de sus coordenadas:

public double obtenerX()

public double obtenerY()

public void cambiarX(double nuevoX)

public void cambiarY(double nuevoY)

2. saber si el punto está sobre el eje de las X:

public boolean estaSobreEjeX()

3. saber si el punto está sobre el eje de las Y:

public boolean estaSobreEjeY()

4. saber si el punto es el origen de coordenadas:

public boolean esElOrigen() { }

5. calcular la distancia al origen y a otro punto:

public double distanciaAlOrigen() { }

public double distanciaAotroPunto(Punto otro) { }

Realizar todas las pruebas que considere convenientes. Se pueden agregar métodos privados.

Ejercicio 03

Implementar una clase que modele un círculo, del que se desea manipular (obtener y cambiar):

- radio
- diámetro
- perímetro
- área

Realizar todas las pruebas que considere convenientes. Se pueden agregar métodos privados.

Ejercicio 04

Implementar la clase Cubo a partir de la interfaz proporcionada.

Realizar todas las pruebas que considere convenientes. Se pueden agregar métodos privados.

Ejercicio 05

Implementar la clase TarjetaBaja a partir de la interfaz proporcionada.

Realizar todas las pruebas que considere convenientes. Se pueden agregar métodos privados.

Ejercicio 06

Implementar la clase Cerradura con los siguientes métodos. Indique axiomas de la clase, pre y post condiciones de los métodos. Cuando una Cerradura se bloquea no puede volver a abrirse nunca más.

```
class Cerradura {
   public Cerradura(int claveDeApertura,
                    int cantidadDeFallosConsecutivosQueLaBloquean)
   public boolean abrir(int clave)
   public void cerrar()
   public boolean estaAbierta()
   public boolean estaCerrada()
   public boolean fueBloqueada()
   public int contarAperturasExitosas()
   public int contarAperturasFallidas()
}
```

Realizar todas las pruebas que considere convenientes. Se pueden agregar métodos privados.

Carrera **INGENIERIA EN INFORMATICA**

Asignatura 3646 - Paradigmas de Programación

Tema Guía de Cuestiones Básicas del Lenguaje Java

Unidad 4

Objetivo Explorar conceptos básicos de la programación en el lenguaje, como toString, final, static, etcétera.

Competencia/s a desarrollar

Genéricas

- **Competencias tecnológicas**

1. Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.
2. Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información / informática.
3. Gestión, planificación, ejecución y control de proyectos de ingeniería en sistemas de información / informática.
4. Utilización de técnicas y herramientas de aplicación en la ingeniería en sistemas de información / informática.

- **Competencias sociales, políticas y actitudinales**

5. Desempeño en equipos de trabajo.
6. Comunicación efectiva.
7. Actuación profesional ética y responsable.
8. Aprendizaje continuo.
9. Desarrollo de una actitud profesional emprendedora.

Descripción de la Actividad

Tiempo estimado de resolución: 90 minutos

Metodología:

Resolución de problemas de programación con una guía autoexplicativa.

Forma de entrega:

Entrega presencial opcional durante el período de clase destinado a tal fin.

Metodología de Corrección y Retroalimentación:

Se realizará una puesta en común de ejercicios seleccionados. Se utilizará el código provisto por estudiantes voluntarios para analizar distintas posibles resoluciones.

Se utilizarán técnicas de pair programming y mob programming para resolver los ejercicios que se seleccionen para hacer en la clase.

Guía de Ejercicios: Cuestiones básicas del lenguaje Java

https://github.com/paradigmas-de-programacion/guia-basicas

Resolver los siguientes ejercicios utilizando el lenguaje de programación Java, con el paradigma imperativo. Encontrarás una estructura de proyecto conveniente para ese objetivo.

Ejercicio 01

Implementar la clase Rango, que representará los conceptos de intervalos numéricos. Un rango puede ser abierto a izquierda, a derecha, abierto o cerrado.

Más información: Wikipedia

Deberá cumplir con la siguiente especificación:

1. Un rango deberá poder crearse en sus cuatro combinaciones posibles:

[1.0, 2.0]

[1.0, 2.0)

(1.0, 2.0]

(1.0, 2.0)

2. Dado que tener un constructor tan complejo puede ser perjudicial, implementar cuatro métodos estáticos que permitan la creación de estas combinaciones.

3. Teniendo los métodos estáticos, será buena idea hacer el constructor privado, ya que sólamente se accederá a él por los métodos estáticos.

4. Se debe poder consultar si un número está dentro de un rango.

5. Se debe poder consultar si un rango está dentro de un rango.

6. Se debe poder consultar si hay intersección entre dos rangos.

7. Se debe poder comparar por igualdad los rangos.

8. Se debe poder ordenar rangos mediante su inicio. Si empatan, se resuelve el empate por su fin. Si empatan, los rangos cerrados irán primero que los abiertos.

9. Se debe poder imprimir un rango en formato cadena de caracteres.

10. Un rango es inmutable: no puede modificarse una vez creado.

11. Proporcionar un método estático que devuelva un rango que abarque a todos los otros rangos.

12. Se deben poder sumar rangos, utilizando el inicio del primero y el fin del segundo.

13. Se debe poder calcular un rango intersección, que en caso de no existir tal intersección retornará `(0.0, 0.0)`

14. Se debe poder desplazar un rango con un número escalar.

Realizar todas las pruebas que considere convenientes. Se pueden agregar métodos privados.

Carrera **INGENIERIA EN INFORMATICA**

Asignatura 3646 - Paradigmas de Programación

Tema Guía de Entrada/Salida

Unidad 4

Objetivo Ejercitar la correcta lectura y escritura de archivos en diferentes formatos, con las herramientas provistas por el lenguaje

Competencia/s a desarrollar

Genéricas

- **Competencias tecnológicas**

1. Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.
2. Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información / informática.
3. Gestión, planificación, ejecución y control de proyectos de ingeniería en sistemas de información / informática.
4. Utilización de técnicas y herramientas de aplicación en la ingeniería en sistemas de información / informática.

- **Competencias sociales, políticas y actitudinales**

5. Desempeño en equipos de trabajo.
6. Comunicación efectiva.
7. Actuación profesional ética y responsable.
8. Aprendizaje continuo.
9. Desarrollo de una actitud profesional emprendedora.

Descripción de la Actividad

Tiempo estimado de resolución: 1 hora

Metodología:
Resolución de problemas de programación con una guía autoexplicativa.

Forma de entrega:
Entrega presencial opcional durante el período de clase destinado a tal fin.

Metodología de Corrección y Retroalimentación:
Se realizará una puesta en común de ejercicios seleccionados. Se utilizará el código provisto por estudiantes voluntarios para analizar distintas posibles resoluciones. Se utilizarán técnicas de pair programming y mob programming para resolver los ejercicios que se seleccionen para hacer en la clase.

Bibliografía

-

Guía de Ejercicios: Entrada/Salida

https://github.com/paradigmas-de-programacion/guia-entrada-salida

Resolver los siguientes ejercicios utilizando el lenguaje de programación Java, con el paradigma imperativo. Encontrarás una estructura de proyecto conveniente para ese objetivo.

Ejercicio 01

Escribir un programa en Java, con todas las clases que considere necesarias, para la lectura y procesamiento de un archivo, de la siguiente manera:

Se recibe un archivo en el cual la primera línea representa la cantidad de enteros que vendrán a continuación, y luego esa cantidad de enteros positivos, entre 0 y 9. Se pide escribir un archivo de salida que contenga en su primera línea un contador de números leídos (sin repetidos) y luego los números ordenados.

Ejemplo 1:

`caso01.in`:

```
7
1
3
2
1
3
8
2
```

`caso01.out`:

```
4
1
2
3
8
```

Ejemplo 2:

`caso02.in`:

```
0
```

`caso02.out`:

```
0
```

Ejercicio 02

Escribir un programa en Java, con todas las clases que considere necesarias, para la lectura y procesamiento de un archivo, de la siguiente manera:

1. Escribir un archivo de entre 10 mil y 20 mil números enteros aleatorios, en el rango de 0 a 12000. (La cantidad de números debe ser aleatoria)

2. Leer dicho archivo, buscando (a) máximo, (b) mínimo, (c) promedio.

3. Escribr un segundo archivo con una tabla de resultados. Ejemplo:

```
+----------+-------+
| Máximo   | 11978 |
+----------+-------+
| Mínimo   |     3 |
+----------+-------+
| Promedio |  7201 |
+----------+-------+
```

¡El formato de la última salida es parte del desafío!

Ejercicio 03

Resolver el ejercicio Luchadores Japoneses, que encontrará en este mismo repositorio bajo el nombre de `sumo.pdf` dentro del paquete correspondiente.

El ejercicio deberá resolverse utilizando las técnicas de lectura y escritura de archivos aprendidas, pero también se solicita que se intente aplicar un diseño orientado a objetos que emplee las técnicas y herramientas aprendidas. En específico:

- Separación de responsabilidades en distintas clases
- Colaboración entre objetos
- Composición/Agregación si fuera necesario

Carrera **INGENIERIA EN INFORMATICA**

Asignatura 3646 - Paradigmas de Programación

Tema Guía de Excepciones

Unidad 3

Objetivo Explora el uso de esta herramienta básica para la gestión de flujos excepcionales en la ejecución del software

Competencia/s a desarrollar

Genéricas

- **Competencias tecnológicas**

1. Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.
2. Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información / informática.
3. Gestión, planificación, ejecución y control de proyectos de ingeniería en sistemas de información / informática.
4. Utilización de técnicas y herramientas de aplicación en la ingeniería en sistemas de información / informática.

- **Competencias sociales, políticas y actitudinales**

5. Desempeño en equipos de trabajo.
6. Comunicación efectiva.
7. Actuación profesional ética y responsable.
8. Aprendizaje continuo.
9. Desarrollo de una actitud profesional emprendedora.

Descripción de la Actividad

Tiempo estimado de resolución: 1 hora

Metodología:
Resolución de problemas de programación con una guía autoexplicativa.

Forma de entrega:
Entrega presencial opcional durante el período de clase destinado a tal fin.

Metodología de Corrección y Retroalimentación:
Se realizará una puesta en común de ejercicios seleccionados. Se utilizará el código provisto por estudiantes voluntarios para analizar distintas posibles resoluciones. Se utilizarán técnicas de pair programming y mob programming para resolver los ejercicios que se seleccionen para hacer en la clase.

Guía de Ejercicios: Excepciones

https://github.com/paradigmas-de-programacion/guia-excepciones

Resolver los siguientes ejercicios utilizando el lenguaje de programación Java, con el paradigma imperativo. Encontrarás una estructura de proyecto conveniente para ese objetivo.

Ejercicio 01

Escribir una función que calcule la división de dos números enteros, pero que lance una excepción del tipo `ArithmeticException` si el segundo número es cero.

Ejercicio 02

Definir su propia excepción y utilizarla en un programa que posea una función que lance dicha excepción si se intenta calcular la raiz cuadrada de un número negativo. Esta excepción deberá extender de `Exception` en forma directa.

Ejercicio 03

Escribir un programa que calcule el cociente de dos números enteros, pero que lance una **excepción propia** de tiempo de ejecución si el segundo número es cero. Tip: Deberá extender de `RuntimeException`.

¿Necesita ser anunciada? (throws)

Ejercicio 04

Escribir una clase que simule una cuenta bancaria. El programa debe lanzar una excepción si se intenta retirar más dinero del que hay disponible en la cuenta.

Ejercicio 05

Agregar excepciones al ejercicio anterior (realizar una copia). El programa debe usar try-catch para manejar cualquier excepción que pueda ocurrir. Para ello, agregaremos al menos dos excepciones nuevas:

1. Crear una cuenta con saldo negativo
2. Retirar/Depositar montos negativos

Ejercicio 06

Escribir un programa que lea un archivo de texto que contiene el nombre de un archivo a su vez. El programa debe intentar abrir y leer el archivo mencionado en el primer archivo, y lanzar una excepción si no se puede abrir el archivo mencionado. Cerrar el primer archivo luego de cerrar el segundo. Esto generará que utilices dos bloques try anidados.

Ejercicio 07

Repetir el ejercicio anterior, pero utilizando bloques try with resources. Comparar el código resultante.

Ejercicio 08

Explorar el uso de la palabra reservada `assert` para verificar ciertas condiciones durante la ejecución del código. Por ejemplo, que el dividendo sea diferente a cero en una división.

Carrera **INGENIERIA EN INFORMATICA**

Asignatura 3646 - Paradigmas de Programación

Tema Guía de Colecciones

Unidad 4

Objetivo Explorar y contrastar el uso de distintas colecciones provistas por el Java Collection Framework

Competencia/s a desarrollar

Genéricas

- **Competencias tecnológicas**

1. Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.
2. Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información / informática.
3. Gestión, planificación, ejecución y control de proyectos de ingeniería en sistemas de información / informática.
4. Utilización de técnicas y herramientas de aplicación en la ingeniería en sistemas de información / informática.

- **Competencias sociales, políticas y actitudinales**

5. Desempeño en equipos de trabajo.
6. Comunicación efectiva.
7. Actuación profesional ética y responsable.
8. Aprendizaje continuo.
9. Desarrollo de una actitud profesional emprendedora.

Descripción de la Actividad

Tiempo estimado de resolución: 2 horas

Metodología:
Resolución de problemas de programación con una guía autoexplicativa.

Forma de entrega:
Entrega presencial opcional durante el período de clase destinado a tal fin.

Metodología de Corrección y Retroalimentación:
Se realizará una puesta en común de ejercicios seleccionados. Se utilizará el código provisto por estudiantes voluntarios para analizar distintas posibles resoluciones. Se utilizarán técnicas de pair programming y mob programming para resolver los ejercicios que se seleccionen para hacer en la clase.

Guía de Ejercicios: Colecciones

https://github.com/paradigmas-de-programacion/guia-colecciones

Resolver los siguientes ejercicios utilizando el lenguaje de programación Java, con el paradigma imperativo. Encontrarás una estructura de proyecto conveniente para ese objetivo.

Listas

Ejercicio 01

Una empresa de mensajería necesita llevar un registro de los paquetes que transporta. Cada paquete tiene un número de seguimiento, dirección de origen, dirección de destino y peso. Escriba un programa que permita:

1. Agregar un nuevo paquete al registro.
2. Buscar un paquete por número de seguimiento.
3. Mostrar la lista de paquetes que superan un peso determinado.

Ejercicio 02

Una tienda de ropa necesita llevar un registro de las ventas que realiza cada día. Cada venta tiene un número de venta, fecha, hora, nombre del cliente y monto de la venta. Escriba un programa que permita:

1. Agregar una nueva venta al registro.
2. Buscar una venta por número de venta.
3. Mostrar la lista de ventas realizadas en un día determinado.

Reflexión

Usted está llegando a clases y escucha a otro estudiante decir "uno de estos ejercicio se resolvía mejor con `LinkedList` y el otro con `ArrayList`". ¿Está de acuerdo? ¿Por qué?

Conjuntos

Ejercicio 03

Una aplicación necesita llevar un registro de las palabras únicas que aparecen en un texto. Diseñe un programa que reciba el texto y utilice un Set para almacenar las palabras únicas que aparecen en él. Devuelva el listado de palabras únicas.

Ejercicio 04

Una tienda de libros necesita llevar un registro de los títulos únicos de los libros que tiene en stock. Diseñe un programa que utilice un Set para almacenar los libros sin ejemplares repetidos.

> Tip: Para esto necesitará definir el criterio de igualdad correctamente. Defina los atributos en consecuencia.

Mapas

Ejercicio 05

Una empresa necesita llevar un registro de las ventas que ha realizado cada mes. Diseñe un programa que utilice un HashMap para almacenar el mes como clave y el monto de ventas como valor. El programa debe permitir agregar nuevas ventas y mostrar el monto de ventas de un mes específico.

Ejercicio 06

Un programa necesita llevar un registro de los estudiantes y sus respectivas notas en un curso. Escriba un programa que utilice un HashMap para almacenar el nombre del estudiante como clave y un arreglo de notas como valor. El programa debe permitir agregar nuevas notas y mostrar el promedio de notas de un estudiante específico.

Ahora diseñe una función que permita invertir el mapa: queremos el listado de estudiantes por promedio. Ante un mismo promedio, debe devolver todos los estudiantes que lo hayan obtenido.

Colas

Ejercicio 07

Una aplicación necesita procesar una lista de tareas en orden de llegada. Diseñe un programa que utilice una cola para almacenar las tareas a medida que se reciben. El programa debe procesar las tareas en el orden en que se recibieron.

Ejercicio 08

Diseñe una función que permita invertir los elementos de una cola sin utilizar ninguna estructura auxiliar para ello. Es decir, luego de la ejecución de esta función, una cola con los elementos {1, 2, 3} deberá tener los elementos {3, 2, 1}.

Pilas

Ejercicio 09

Un programa necesita determinar si una expresión aritmética es válida. Diseñe un programa que utilice una pila para almacenar los paréntesis abiertos y cerrados en la expresión. El programa debe asegurarse de que los paréntesis estén correctamente balanceados.

Ejercicio 10

Un programa necesita procesar una serie de comandos de usuario que se reciben en una terminal. Escriba un programa que utilice una pila para almacenar los comandos a medida que se reciben. El programa debe permitir deshacer la última operación realizada por el usuario desapilando el último comando almacenado.

> Tip: Los comandos serán simulados, no deben ejecutarse realmente. Conviene modelar la clase `Terminal` con sus métodos para introducir comandos y deshacerlos.

Carrera **INGENIERIA EN INFORMATICA**

Asignatura 3646 - Paradigmas de Programación

Tema Guía de Herencia y Polimorfismo

Unidad 3

Objetivo Acentúa el buen uso de un diseño jerárquico y polimórfico.

Competencia/s a desarrollar
Genéricas

- **Competencias tecnológicas**

1. Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.
2. Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información / informática.
3. Gestión, planificación, ejecución y control de proyectos de ingeniería en sistemas de información / informática.
4. Utilización de técnicas y herramientas de aplicación en la ingeniería en sistemas de información / informática.

- **Competencias sociales, políticas y actitudinales**

5. Desempeño en equipos de trabajo.
6. Comunicación efectiva.
7. Actuación profesional ética y responsable.
8. Aprendizaje continuo.
9. Desarrollo de una actitud profesional emprendedora.

Descripción de la Actividad

Tiempo estimado de resolución: 2 horas

Metodología:
Resolución de problemas de programación con una guía autoexplicativa.

Forma de entrega:
Entrega presencial opcional durante el período de clase destinado a tal fin.

Metodología de Corrección y Retroalimentación:
Se realizará una puesta en común de ejercicios seleccionados. Se utilizará el código provisto por estudiantes voluntarios para analizar distintas posibles resoluciones. Se utilizarán técnicas de pair programming y mob programming para resolver los ejercicios que se seleccionen para hacer en la clase.

Guía de Ejercicios: Herencia y polimorfismo

https://github.com/paradigmas-de-programacion/guia-herencia

Resolver los siguientes ejercicios utilizando el lenguaje de programación Java, con el paradigma imperativo. Encontrarás una estructura de proyecto conveniente para ese objetivo.

Ejercicio 01: Ejercicio compuesto de cuentas bancarias

El siguiente es un ejercicio compuesto de varias partes. Es un compilado para visitar y profundizar en los conceptos de Programación Orientada a Objetos. Se anima a los estudiantes a explorar el ejercicio, codificarlo y expandirlo cuanto les parezca necesario. Mediante la exploración es que se lograrán los aprendizajes más profundos, se debe tomar este ejercicio como un mapa: marca el camino, pero si nos desviamos para descubrir nuevas áreas, siempre podemos retomar sendero gracias a él.

Se recomienda agregar los tests necesarios para poder comprobar que el comportamiento deseado es obtenido, y que al continuar a la siguiente parte, todo lo anterior siga funcionando de manera correcta

Parte 1: Estructuras de Datos

Una Cuenta posee un saldo. Se puede agregar dinero a la Cuenta, incrementando el saldo. También se puede retirar dinero de la Cuenta, decrementando dicho saldo. Debe poderse resolver la siguiente secuencia de acciones:

- Creación de una Cuenta nueva, saldo == 0.
- Acreditación de $ 1000, saldo == 1000.
- Retiro de $ 550, saldo == 450.

```
// main

Cuenta miCuenta = new Cuenta();

System.out.println(miCuenta.saldo); // 0

miCuenta.saldo += 1000;

System.out.println(miCuenta.saldo); // 1000

miCuenta.saldo -= 550;

System.out.println(miCuenta.saldo); // 450
```

Preguntas para profundizar

- ¿Qué tipo de dato se utilizó para el saldo? ¿Es el más apropiado?
- ¿Qué visibilidad tiene el atributo saldo de la Cuenta? ¿Es correcto?
- Si extraigo más dinero del disponible en saldo, seguramente pueda hacerlo. ¿Está bien que así sea?

Parte 2: Encapsulamiento

Hemos visto las falencias del diseño anterior, y luego de investigar un poco nos encontramos con el concepto de Encapsulamiento. ¡Es una buena ocasión para incorporarlo! Se pide que utilicemos este concepto para asegurarnos de las siguientes restricciones del dominio del problema:

- No puedo extraer dinero que no poseo
- No puedo extraer ni depositar dinero negativo
- El saldo no puede afectarse más allá que por depósitos o extracciones

Preguntas para profundizar

- ¿Qué ventaja representó la incorporación del Encapsulamiento en el problema?
- ¿Qué desventaja podríamos encontrar?
- ¿Es realmente Programación Orientada a Objetos? ¿Cómo puedo saberlo?
- ¿Qué acciones se utilizaron para prevenir la realización de operaciones no válidas?

Parte 3: Abriendo el juego

Es común transferir dinero entre cuentas bancarias. Para ello, debemos dotar a nuestro modelo con dicha funcionalidad. Cuando querramos transferir un monto, deberá verificarse que la operación puede realizarse, y deberá acreditarse directamente en la cuenta destino. El código para dicha operación se verá similar a este:

```
// main

Cuenta cuentaOrigen = new Cuenta();

cuentaOrigen.depositar(10000);

System.out.println(cuentaOrigen.consultarSaldo()); // 10000

Cuenta cuentaDestino = new Cuenta();

System.out.println(cuentaDestino.consultarSaldo()); // 0

cuentaOrigen.transferir(550, cuentaDestino);

System.out.println(cuentaOrigen.consultarSaldo()); // 9450

System.out.println(cuentaDestino.consultarSaldo()); // 550
```

Preguntas para profundizar

- Al interactuar con objetos del mismo tipo, es importante distinguir entre parámetros y atributos. ¿Cómo puede hacerse para evitar confusiones?
- ¿Es correcto el orden de los parámetros para el método transferir? ¿Qué cambiarías?
- Dentro del código de transferir, ¿utilizaste los métodos preexistentes o repetiste lógica que ya existía? ¿Cuál es la ventaja de cada una de las aproximaciones?
- ¿Qué pasaría si la operación de transferir se ve interrumpida a la mitad de su ejecución? ¿Cómo se podría prevenir esto?

Parte 4: Se agranda la familia

Esta vez necesitamos modelar dos nuevos tipos de cuenta:

- CuentaDeAhorros, que permite reservar parte del saldo para que no esté disponible, en una especie de saldo secundario. Se puede disponer de ese saldo normalmente una vez que se reintegre (a pedido del cliente) al saldo total de la cuenta.
- CuentaCorriente, que permite retirar todo lo disponible, y un descubierto de hasta cierta cantidad de dinero extra. El monto “en descubierto” se establece al momento de la apertura de la cuenta.

Preguntas para profundizar

- ¿Hubo alguna funcionalidad que pudiera reutilizarse?
- ¿Qué método tuvo que rehacerse?
- ¿Cómo se manejan transferencias entre tipos de cuenta diferentes?

Parte 5: Policomplicaciones

Parte 5.1:

El banco decide comenzar a registrar las transacciones individualmente, por lo que necesitaremos mantener un registro de cada acreditación o débito en la cuenta. Esto debe suceder para todos los tipos de cuenta. De la transacción se desea conocer el motivo y el monto. Es importante mantenerlas ordenadas. No es importante saber la fecha en la que se realizó, pero sería bueno hacerlo.

Parte 5.2:

Ahora que tenemos registro de transacciones, podemos incorporar nuevos productos bancarios:

- Tarjetas de Débito, las cuales tienen una cuenta asociada y retiran dinero de ella al efectuar una compra.
- Tarjetas de Crédito, las cuales permiten acumular compras hasta el momento en el que finalmente se debitan de la cuenta, donde se cobra el total más una comisión del 3% sobre el total comprado en concepto de gastos administrativos.
- Plazos Fijos, que nos privan de cierto monto de dinero, pero al momento de acreditarse se reintegran con un interés adicional al capital reservado. El interés actualmente es del 36% anual, pero vamos a integrar mensualmente.

Ejercicio 02: Instrumentos musicales

Se quiere construir un programa que permita al usuario utilizar instrumentos musicales. El sistema tendrá las siguientes características:

- Se podrá consultar la lista de todos los instrumentos musicales que contiene el programa.
- Todos los instrumentos tienen un nombre y una descripción.
- Todos los instrumentos se pueden tocar. Pero el funcionamiento de tocar un instrumento varía dependiendo del instrumento que sea.
- Los instrumentos se clasifican en 3 tipos: Viento, Percusión y Cuerda. A su vez los instrumentos de viento pueden ser de 2 tipos: Madera o Metal.
- Los instrumentos de cuerda y de viento son afinables. Y hay dos maneras de afinar un instrumento: en forma manual o en forma automática. El funcionamiento de cada forma de afinación varía de acuerdo al tipo de instrumento.
- Los instrumentos de percusión y de viento son lustrables. Y el funcionamiento del lustrado de un instrumento varía dependiendo del tipo de instrumento.

Nota: Cuando realice el código de las clases, que la implementación de los métodos sólo sea una impresión en pantalla el nombre del método que se está ejecutando. Se pide realizar un Main para cada uno de estos objetivos:

- Crear una orquesta, compuesta por varios instrumentos. Debe ser posible tocar todos los instrumentos juntos, o los de viento, percusión y cuerdas por separado.
- Crear un "Lustrador", que pueda lustrar instrumentos lustrables.
- Crear un "Afinador", que pueda afinar instrumentos afinables.

Carrera **INGENIERIA EN INFORMATICA**

Asignatura 3646 - Paradigmas de Programación

Tema Guía de Patrones de Diseño

Unidad 4

Objetivo Explorar el uso de algunos patrones de diseño significativos

Competencia/s a desarrollar

Genéricas

- **Competencias tecnológicas**

1. Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.
2. Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información / informática.
3. Gestión, planificación, ejecución y control de proyectos de ingeniería en sistemas de información / informática.
4. Utilización de técnicas y herramientas de aplicación en la ingeniería en sistemas de información / informática.

- **Competencias sociales, políticas y actitudinales**

5. Desempeño en equipos de trabajo.
6. Comunicación efectiva.
7. Actuación profesional ética y responsable.
8. Aprendizaje continuo.
9. Desarrollo de una actitud profesional emprendedora.

Descripción de la Actividad

Tiempo estimado de resolución: 90 minutos

Metodología:
Resolución de problemas de programación con una guía autoexplicativa.

Forma de entrega:
Entrega presencial opcional durante el período de clase destinado a tal fin.

Metodología de Corrección y Retroalimentación:
Se realizará una puesta en común de ejercicios seleccionados. Se utilizará el código provisto por estudiantes voluntarios para analizar distintas posibles resoluciones. Se utilizarán técnicas de pair programming y mob programming para resolver los ejercicios que se seleccionen para hacer en la clase.

Guía de Ejercicios: Patrones de Diseño

https://github.com/paradigmas-de-programacion/guia-patrones

Resolver los siguientes ejercicios utilizando el lenguaje de programación Java, con el paradigma imperativo. Encontrarás una estructura de proyecto conveniente para ese objetivo.

Ejercicio 01: Composite

Augusto quiere ayudar a su hija con una tarea del jardín de infantes. Le han pedido que lleve materiales para pintar una serie de formas geométricas con colores. Habrá un grupo de formas que pintará de un color, y otro grupo que pintará de otro color, y así sucesivamente.

Ella dispone por anticipado de las medidas y ubicaciones precisas de cada figura, pero necesita saber cuántos pomos de cada témpera debe comprar. Sabiendo que un pomo de témpera tiene un poder cubritivo de 100cm2, y que no puede comprar "fracciones de pomo", se te pide que le muestres una forma de calcular las cantidades a comprar utilizando el patrón de diseño Composite.

> **Nota:** Las figuras no se solapan entre sí. Cada figura estará pintada al 100% del color indicado.

Ejercicio 02: State

Considere que tiene una clase llamada "Tank" que representa a un tanque Terran en Starcraft 2. Implemente el patrón State para modelar los dos posibles estados del tanque: el estado "Modo Tanque" y el estado "Modo Asedio". A continuación, se proporcionan varios métodos que puede incluir en la implementación:

- Tank: la clase principal que representa un tanque Terran y contiene una referencia a un objeto de estado concreto.

- TankState: la interfaz que define los métodos comunes que deben implementar los estados concretos.

- TankModeTankState: una clase que implementa la interfaz TankState y representa el estado "Modo Tanque" del tanque. Debe proporcionar implementaciones para los métodos específicos de este estado, como moverse() y atacar().

- TankModeSiegeState: una clase que implementa la interfaz TankState y representa el estado "Modo Asedio" del tanque. Debe proporcionar implementaciones para los métodos específicos de este estado, como moverse() y atacar().

Carrera **INGENIERIA EN INFORMATICA**

Asignatura 3646 - Paradigmas de Programación

Tema Guia de Prolog

Unidad 5

Objetivo Reforzar y ejercitar las técnicas principales de la programación lógica con el lenguaje Prolog

Competencia/s a desarrollar

Genéricas

- **Competencias tecnológicas**

1. Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.
2. Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información / informática.
3. Gestión, planificación, ejecución y control de proyectos de ingeniería en sistemas de información / informática.
4. Utilización de técnicas y herramientas de aplicación en la ingeniería en sistemas de información / informática.

- **Competencias sociales, políticas y actitudinales**

5. Desempeño en equipos de trabajo.
6. Comunicación efectiva.
7. Actuación profesional ética y responsable.
8. Aprendizaje continuo.
9. Desarrollo de una actitud profesional emprendedora.

Descripción de la Actividad

Tiempo estimado de resolución: 4 horas

Metodología:
Resolución de problemas de programación con una guía autoexplicativa.

Forma de entrega:
Entrega presencial opcional durante el período de clase destinado a tal fin.

Metodología de Corrección y Retroalimentación:
Se realizará una puesta en común de ejercicios seleccionados. Se utilizará el código provisto por estudiantes voluntarios para analizar distintas posibles resoluciones.
Se utilizarán técnicas de pair programming y mob programming para resolver los ejercicios que se seleccionen para hacer en la clase.

Guía de Ejercicios: Prolog

https://github.com/paradigmas-de-programacion/guia-prolog

Resolver los siguientes ejercicios utilizando el lenguaje de programación Prolog, con el paradigma lógico. Encontrarás una estructura de proyecto conveniente para ese objetivo.

Ejercicio 01

Las siguientes cláusulas corresponden al programa "menú" de un restaurante. El restaurante ofrece menús completos compuestos por una entrada, un plato principal y un postre. El plato principal puede ser carne o pescado.

entrada(paella).

entrada(gazpacho).

entrada(consomé).

carne(filete_de_cerdo).

carne(pollo_asado).

pescado(trucha).

pescado(bacalao).

postre(flan).

postre(helado).

postre(pastel).

Implementar las reglas necesarias para formular las siguientes consultas en Prolog:

1. ¿Cuáles son los menús que ofrece el restaurante?
2. ¿Cuáles son los menús que tienen Consomé en las entradas?
3. ¿Cuáles son los menús que no contienen flan como postre?

Ejercicio 02

Completar el programa "menú" de manera que una comida esté formada también por la elección de una bebida, a elegir entre vino, cerveza o agua mineral.

Ejercicio 03

El árbol genealógico siguiente se describe con el programa Prolog:

hombre(pedro) .

hombre(manuel) .

hombre(arturo) .

mujer(maría).

padre(pedro, manuel).

padre(pedro, arturo).

padre(pedro, maría) .

A partir de estas afirmaciones, formular las reglas generales de:

niño(X,Y) % expresa que X es hijo o hija de Y.

hijo(X,Y) % expresa que X es un hijo varón de Y.

hija(x,y) % expresa que X es una hija de Y.

hermano-o-hermana(X,Y) % expresa que X es hermano o hermana de Y.

hermano(X,Y) % expresa que X es un hermano de Y.

hermana(X,Y) % expresa que X es una hermana de Y.

Nota: Un individuo no puede ser hermano ni hermana de sí mismo.

Ejercicio 04

Una agencia de viajes propone a sus clientes viajes de una o dos semanas a Roma, Londres o Túnez. El catálogo de la agencia contiene, para cada destino, el precio del transporte (con independencia de la duración) y el precio de una semana de estancia que varía según el destino y el nivel de comodidad elegidos: hotel, hostal o camping.

Escribir el conjunto de declaraciones que describen este catálogo (se muestra a continuación).

transporte(roma,20).

transporte(londres,30).

transporte(tunez,10).

alojamiento(roma,hotel,50).

alojamiento(roma,hostal,30).

alojamiento(roma,camping,10).

alojamiento(londres,hotel,60).

alojamiento(londres,hostal,40).

alojamiento(londres,camping,20).

alojamiento(tunez,hotel,40).

alojamiento(tunez,hostal,20).

alojamiento(tunez,camping,5).

1. Expresar la relación `viaje(C,S,H,P)` que se interpreta por: "el viaje a la ciudad `C` durante `S` semanas con estancia en `H` cuesta `P` pesos"

2. Completar con `viajeeconomico(C,S,H,P,Pmax)` que expresa que el costo `P` es menor que `Pmax` pesos.

Ejercicio 05

Se dispone de un listado con los resultados de los parciales de los alumnos del curso de **Paradigmas de Programación**, organizado de la siguiente manera:

Versión 1

Se dispone de las notas de ambos parciales para todos los alumnos.

parcial1(ana,7).

parcial1(juan,4).

.

.

.

parcial2(ana,9).

parcial2(juan,8).

.

.

.

Se desea obtener:

1. El listado de los alumnos que promocionan la materia, indicando el nombre y la nota final (promedio de los dos parciales), para cada uno.

2. El listado de los alumnos que obtendrán la cursada.

3. El listado de los alumnos que recursan la materia.

4. A fin de entregar la medalla al mérito, encontrar de aquellos que promocionan, el o los alumnos con mayor nota final.

Versión 2

El listado se compone de al menos una nota para cada alumno. En caso de que algún alumno adeude uno de los parciales, su condición es ausente. No se toman en cuenta quienes no dieron ninguno de los parciales.

Ejercicio 06

Una agencia matrimonial de los años '80 tiene un fichero de candidatos al matrimonio organizado según las declaraciones siguientes:

hombre(N,A,C,E).

mujer(N,A,C,E).

Donde N es el nombre de un hombre o una mujer, A su altura (alta, media, baja), C el color de su cabello (rubio, castaño, pelirrojo, negro) y E su edad (joven,adulta,vieja).

gusta(N,M,L,S).

Que indica que a la persona N le gusta el género de música M (clásica, pop, jazz), el género de literatura L (aventura, ciencia-ficción, policíaca), y practica el deporte S (tenis, natación, jogging).

busca(N,A,C,E).

Que expresa que la persona N busca una pareja de altura A, con cabello color C y edad E.

Se considera que dos personas x e y de sexos diferentes son adecuadas si x conviene a y e y conviene a x. Se dice que x conviene a y si x conviene físicamente a y (la altura, edad y color de cabello de y son los que busca x) y si, además, los gustos de x e y en música, literatura y deporte coinciden.

Ejercicio 07

Con el programa 'comida', dado en clase, describir la semántica de las siguientes tres consultas Prolog y decir cuál es el resultado de la ejecución:

comida(E,P,D),!.

comida(E,P,D),pescado (P),!.

comida(E,P,D),!,pescado(P).

Analizar el comportamiento del operador ! (operador de corte o " cut").

Ejercicio 08

Modificar el programa del **Ejercicio 01** (menú) para poder consultar cual es el menú completo que tiene menor cantidad de calorías.

Ejercicio 09

Basado en el ejemplo de paises.pl visto en clase, complete la base de conocimientos `pais_superficie(P,A)` con todos los países de latinoamérica y codifique las reglas prolog que permitan encontrar el país de mayor superficie.

Ejercicio 10

Dado el listado de vendedores y ventas semestrales se desea obtener el listado anual de comisiones. Las comisiones se liquidan de la siguiente manera:

- 20% del total vendido en el año para aquellos vendedores que hayan tenido ventas en ambos semestres y cada una de ellas supera los $ 20000.
- 10% del total vendido en el año para aquellos vendedores que hayan tenido ventas en ambos semestres, pero no superan los $ 20000 en alguno de estos.
- 5% del total vendido para los vendedores que no registran ventas en algún semestre

Se dispone de los siguientes datos:

ventas1erSem(vendedor, importe).

.

.

ventas2doSem(vendedor, importe).

> **Nota:** No todos los vendedores venden en ambos semestres, todos los importes son mayores que cero. En caso de no registrarse ventas en algún semestre, no figura la regla correspondiente para ese vendedor.

Ejercicio 11: Recursividad

1. Codifique en prolog las reglas necesarias para obtener el término N en la serie de Gauss

2. Codifique en prolog las reglas necesarias para obtener el término N en la serie de Fibonacci

3. Codifique en prolog las reglas necesarias para obtener el factorial de un número natural N.

4. Codifique en prolog las reglas necesarias para obtener el producto de dos numeros X e Y, aplicando sumas sucesivas.

5. Codifique en prolog las reglas necesarias para obtener la potencia N de un numero X aplicando multiplicaciones sucesivas.

6. Codifique en prolog las reglas necesarias para obtener la el cociente entre dos números a partir de restas sucesivas.

7. Idem 6, pero que permita obtener el cociente y el resto. Definir la relación `mcd(X,Y,Z)` que se verifique si Z es el máximo común divisor entre X e Y. Por ejemplo:

mcd(10,15,X).

> X = 5

8. Define un predicado `mcm(X,Y,M)` que signifique "M es el mínimo común múltiplo de X e Y"

Carrera **INGENIERIA EN INFORMATICA**

Asignatura 3646 - Paradigmas de Programación

Tema Guia de Haskell

Unidad 6

Objetivo Conocer y ejercitar técnicas de programación funcional en el lenguaje Haskell

Competencia/s a desarrollar

Genéricas

- **Competencias tecnológicas**

1. Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.
2. Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información / informática.
3. Gestión, planificación, ejecución y control de proyectos de ingeniería en sistemas de información / informática.
4. Utilización de técnicas y herramientas de aplicación en la ingeniería en sistemas de información / informática.

- **Competencias sociales, políticas y actitudinales**

5. Desempeño en equipos de trabajo.
6. Comunicación efectiva.
7. Actuación profesional ética y responsable.
8. Aprendizaje continuo.
9. Desarrollo de una actitud profesional emprendedora.

Descripción de la Actividad

Tiempo estimado de resolución: 4 horas

Metodología:
Resolución de problemas de programación con una guía autoexplicativa.

Forma de entrega:
Entrega presencial opcional durante el período de clase destinado a tal fin.

Metodología de Corrección y Retroalimentación:
Se realizará una puesta en común de ejercicios seleccionados. Se utilizará el código provisto por estudiantes voluntarios para analizar distintas posibles resoluciones. Se utilizarán técnicas de pair programming y mob programming para resolver los ejercicios que se seleccionen para hacer en la clase.

Guía de Ejercicios: Haskell

https://github.com/paradigmas-de-programacion/guia-haskell

Resolver los siguientes ejercicios utilizando el lenguaje de programación Haskell, con el paradigma funcional. Encontrarás una estructura de proyecto conveniente para ese objetivo.

Compilado 01: Consignas Alfa

1. Dado dos números enteros A y B, implemente una función que retorne la división entera de ambos por el método de las restas sucesivas

2. Escribir una función para hallar la potencia de un número

3. Definir una función menor que devuelve el menor de sus dos argumentos enteros

4. Definir una función maximoDeTres que devuelve el máximo de sus argumentos enteros

5. Implemente una función recursiva para calcular el factorial de un número

Compilado 02: Consignas Omega

1. Escribir una función que sume dos números enteros.
2. Implementar una función que calcule el área de un círculo dado su radio.
3. Definir una función que determine si un número es par o impar.
4. Escribir una función que calcule el factorial de un número.
5. Implementar una función que invierta una lista.
6. Definir una función que determine si una lista está ordenada de forma ascendente.
7. Escribir una función que cuente la cantidad de elementos en una lista.
8. Implementar una función que obtenga los elementos en posiciones pares de una lista.
9. Definir una función que calcule el máximo común divisor de dos números.
10. Implementar una función que calcule la suma de los dígitos de un número entero.
11. Definir una función que encuentre el elemento mínimo en una lista.
12. Escribir una función que obtenga el enésimo número de la secuencia de Fibonacci.
13. Implementar una función que verifique si una cadena de texto es un palíndromo.
14. Definir una función que elimine los duplicados de una lista.
15. Implementar una función que obtenga el producto de todos los elementos de una lista.