Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

PRÁCTICA 1:

PAC MAN

PONDERACIÓN: 10 pts

Tiempo estimado: 20 hrs/min

Índice

1. MARCO FORMATIVO

1.1. Valor

Nombre del valor	¿Cómo se aplica en tu laboratorio?
Disciplina	El estudiante sigue el cronograma de forma disciplinada para poder cumplir exitosamente con la entrega en la fecha estipulada.

1.2. Competencia(s)

Tipo de Competencia	
Competencia General	Comunica soluciones técnicas a través de código fuente organizado, un ejecutable funcional (.jar) y el seguimiento de las instrucciones de entrega en el repositorio.
Competencia Específica	Implementa una solución de software en consola utilizando el lenguaje de programación Java, aplicando sentencias de control, ciclos y el manejo de vectores (arreglos) para construir un algoritmo que resuelva un problema práctico.

1.3. Objetivo SMART

SMART	Definición	Objetivo redactado
Específico (¿Qué?)	El objetivo es concreto y tangible.	Aplicar los conocimientos adquiridos en el curso a la solución de un problema práctico a través de una solución de software para un juego básico de consola (Pac-Man).
Medible (¿Cuánto?)	El objetivo tiene una medida objetiva de éxito.	Completar el 100% de las funcionalidades de la aplicación: Menú de Inicio (Iniciar, Historial, Salir), la dinámica del juego (movimiento, puntaje, vidas) y el historial de partidas.
Alcanzable (¿Cómo?)	El objetivo debe ser posible con los recursos disponibles.	Utilizando el lenguaje Java, una aplicación de consola, y únicamente las librerías permitidas (Scanner, Random), aplicando sentencias de control, ciclos y vectores.

Realista (¿Para qué?)	El objetivo contribuye a metas más amplias.	Identificar una solución para un problema a través de su análisis y comprensión, aplicando conceptos de algoritmos a un problema práctico en un entorno real.
A Tiempo (¿Cuándo?)	El objetivo tiene fecha límite o mejor aún un cronograma de hitos de progreso	Cumplir este aprendizaje en dos semanas, las asignadas para está actividad.

2. Enunciado de la Práctica

2.1 Descripción del problema a resolver

Esta práctica consiste en la realización de un juego en consola, utilizando el lenguaje de programación JAVA. Se trata de una versión simple del conocido videojuego PAC MAN, con una interfaz simple pero amigable.

El juego se desarrollará en un tablero de tamaño variable, donde el jugador (PACMAN) deberá moverse para recolectar premios (simples y especiales) y evitar trampas (fantasmas), mientras esquiva paredes.

El objetivo es aplicar los conocimientos adquiridos en el curso, como sentencias de control, ciclos y vectores, en la solución de un problema práctico.

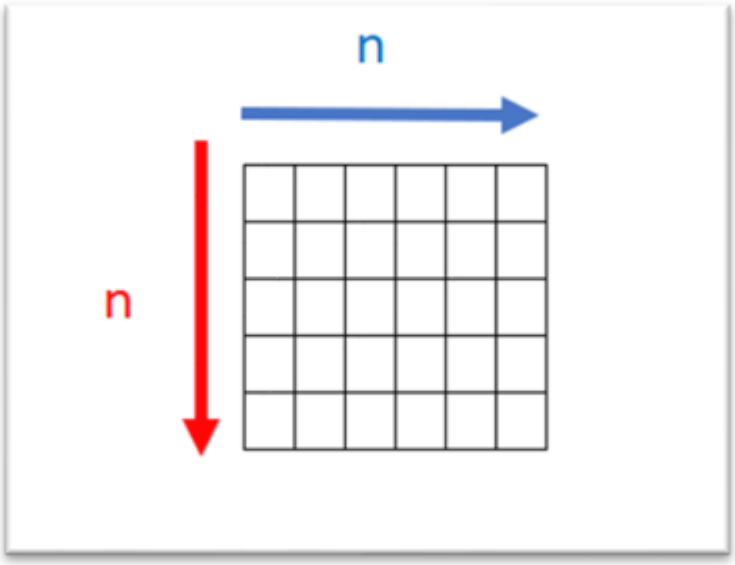
2.2 Alcance de la práctica

2.2 Alcance de la práctica

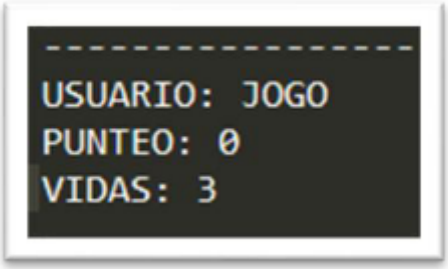
El sistema debe incluir los siguientes componentes, menús y funcionalidades:

Componentes del Juego

- **Tablero:**
  - Será el espacio donde se desarrolla el juego, con forma de tabla (filas y columnas).



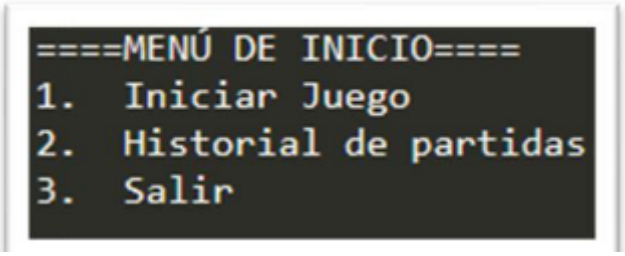
    - El usuario definirá el tamaño antes de la partida, eligiendo entre:
        - **Pequeño:** 5 filas x 6 columnas.
        - **Grande:** 10 filas x 10 columnas.
- **Panel de Control:**
    - Se deben mostrar en todo momento durante la partida los siguientes valores: Nombre de usuario, Punteo, Vidas.



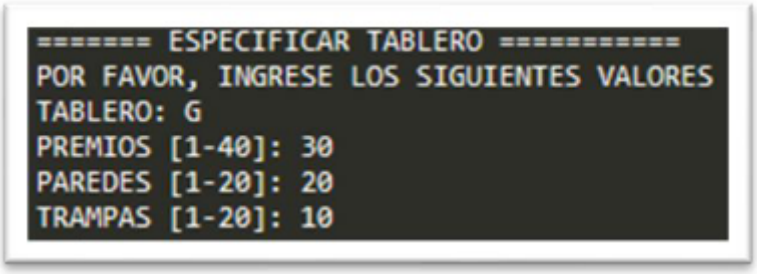
- **Items:**
    - Elementos que interactúan dentro del tablero:
        - **Fantasma:** @
        - **Premio simple:** 0 (Número cero)
        - **Premio especial:** $ (Signo de dólar)
        - **Pared:** X (Equis mayúscula)
        - **PACMAN:** < (Menor que)

Menú de Inicio

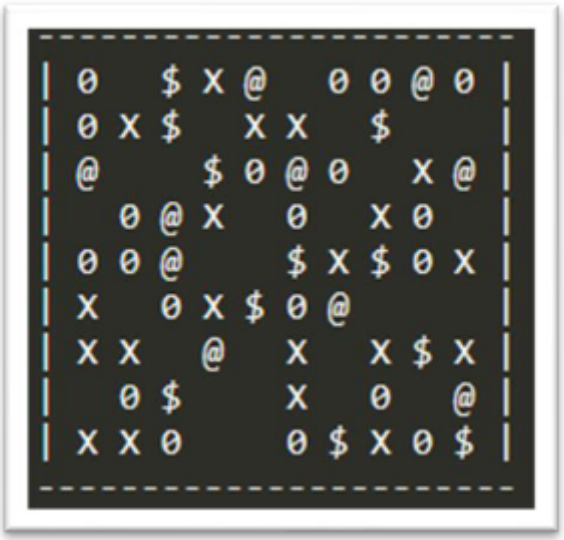
- El menú de inicio contará con 3 opciones: 1. Iniciar Juego, 2. Historial de partidas, 3. Salir.



- **Iniciar Juego:**
  - **Información Principal:** Se debe solicitar el **Nombre de usuario**.
  - **Especificar Tablero:** Se solicita al usuario:
    - **Tipo de tablero:** 'P' (pequeño) o 'G' (grande).
    - **Cantidad de premios:** Mínimo 1, máximo 40% del total de espacios.
    - **Cantidad de paredes:** Mínimo 1, máximo 20% del total de espacios.
    - **Cantidad de trampas (Fantasmas):** Mínimo 1, máximo 20% del total de espacios.

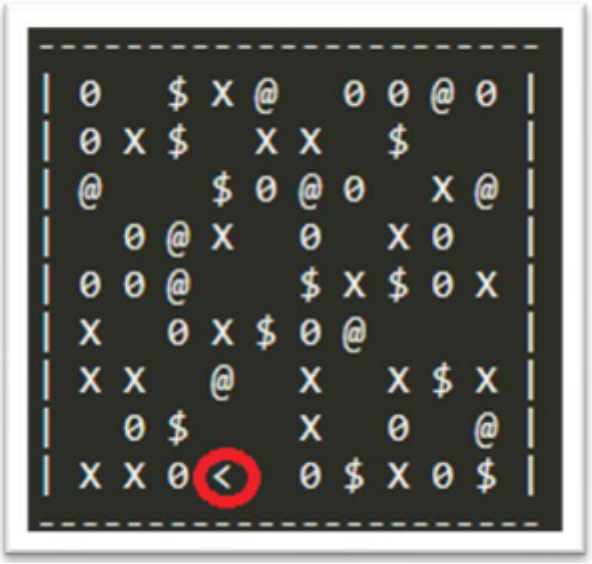


  - Los premios, paredes y trampas se distribuyen aleatoriamente.
  - El tablero generado se le muestra al jugador.



  - Finalmente, se le pide al usuario la **fila y columna** donde desea posicionar su Pac-Man.

```
====== ESPECIFICAR TABLERO ==========
POR FAVOR, INGRESE LA POSICIÓN INICIAL DE JUEGO
FILA: 10
COLUMNA: 4
```



- **Historial de Partidas:**
  - El usuario podrá visualizar un listado de las partidas que ha realizado, incluyendo **Nombre de usuario** y **Puntos**.
  - El orden debe ser del más reciente al más antiguo.

```
==========HISTORIAL DE PARTIDAS==========
No.  USUARIO        PUNTEO

1.   SHROUD          100
2.   NINJA           20
3.   JOGO            15
```

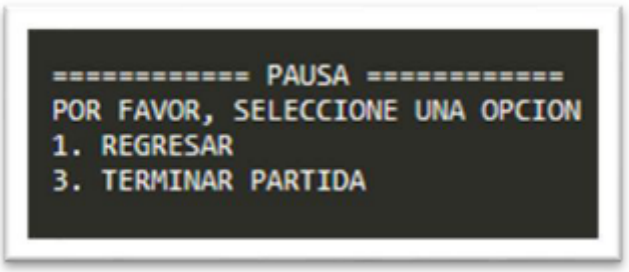
- **Salir:**
  - Finaliza el flujo de la aplicación.
  - La información acumulada (historial) se borrará (no requiere almacenamiento persistente).

Dinámica del Juego

- **Reglas:**
  - El jugador inicia con **3 vidas** y el **puntaje en 0**.

- **Gana** si obtiene todos los premios.
- **No hay bordes externos:** Si Pac-Man se pasa del extremo derecho, reaparece en el izquierdo (y viceversa); igual aplica para arriba y abajo.
- No es posible atravesar las **Paredes (X)**.
- Al pasar por un **Fantasma (@)**, el jugador pierde una vida y el fantasma se borra.
- Al pasar sobre un **Premio (0 o $)**, el jugador gana puntos y el premio se borra.
- El juego termina si: el jugador gana, pierde todas sus vidas, o pausa y termina la partida.
- Al terminar, se almacenan los datos (Usuario, Puntos) para el historial.

- **Movimientos:**
  - Se usarán las siguientes teclas:
    - **8:** ARRIBA
    - **5:** ABAJO
    - **6:** DERECHA
    - **4:** IZQUIERDA
- **Puntaje:**
  - Premio simple (0): **10 puntos**
  - Premio especial ($): **15 puntos**
- **Pausar Juego:**
  - Se detiene el juego presionando la tecla **'F'**.
  - Aparece un menú de pausa con las opciones:
    3. Regresar (regresa a la partida)
    4. Terminar partida (regresa al menú de inicio, registra la partida)

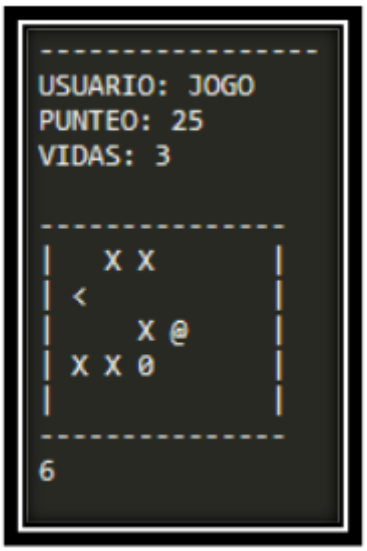


- **Ejemplo de Partida:**


USUARIO: JOGO
PUNTEO: 0
VIDAS: 3
6


USUARIO: JOGO
PUNTEO: 15
VIDAS: 3
6


USUARIO: JOGO
PUNTEO: 25
VIDAS: 3
6



2.3 Requerimientos técnicos

- La aplicación es de Consola (no se requiere de una interfaz gráfica).
- No se permite el uso de arraylist, LinkedList u otra librería con arreglos dinámicos.
- Se deben usar vectores (arreglos) nativos.
- Las copias totales o parciales tendrán una nota de 0. Se permite el uso de las librerías: Scanner y Random.
- El programa se desarrollará en lenguaje Java.

3. Entregables

Describe los productos concretos que se espera que los estudiantes entreguen al finalizar la práctica. Pueden ser prototipos, informes técnicos, documentación, etc.

Ejemplo Software – Introducción a la Programación 1:

Tipo	Descripción
Repositorio en GitHub	Repositorio con el nombre **IPC1<SECCIÓN>_2S2026** que contenga el código fuente de la práctica en una carpeta *Practica1* y un archivo README.md con instrucciones de ejecución.
Código Fuente	Implementación completa en **Java**, organizada y comentada según buenas prácticas de programación.
Manual Técnico	Documento en PDF con la descripción de los métodos utilizados, los requerimientos de la aplicación y observaciones técnicas relevantes.

Manual de Usuario	Documento en PDF con instrucciones claras para interactuar con el sistema, incluyendo ejemplos de uso y capturas de pantalla.
Diagrama de Flujo	Representación gráfica del funcionamiento del sistema y los procesos principales.
Informe de Desarrollo	Documento en PDF que explique la implementación de la práctica, los problemas encontrados y las soluciones adoptadas, con capturas del menú en funcionamiento.

4. Material de apoyo

Colocar material de apoyo que el estudiante puede consultar para la elaboración de la práctica. Incluye links , libros , videos. Todo lo que pueda apoyar al estudiante a lograr el desarrollo de la práctica.

Ejemplo Software – Introducción a la Programación 1:

- Deitel, P., & Deitel, H. (2016). *Cómo programar en Java (10ª edición)*. Pearson Education.
- Joyanes, L. *Java 2 – Manual de Programación*. McGrawHill.
- Documentación oficial de Java SE
- Tutoriales de W3Schools – Java

5. Recursos y herramientas a utilizar

Listado de materiales que los estudiantes deberán usar o investigar:

Ejemplo Software – Introducción a la Programación 1:

- **Software:** Java JDK, NetBeans o IntelliJ IDEA.
- **Plataformas:** GitHub (para repositorio), UEDI (para entrega).
- **Lecturas recomendadas:** manuales de Java, artículos sobre estructuras de datos básicas y diagramas de flujo.

6. Cronograma

El cronograma describe las etapas clave de la práctica, los plazos estimados para cada una, y el proceso de asignación, elaboración y calificación de las tareas. Los estudiantes deberán seguir este plan para asegurar que el práctica avance de manera organizada y cumpla con los plazos establecidos. Cada fase incluye la asignación de tareas, el tiempo estimado para su elaboración, y el momento de su calificación.

Tipo	Fecha Inicio	Fecha Fin
Asignación de Práctica	07/02/2026	07/02/2026
Elaboración	07/02/2026	20/02/2026
Calificación	21/02/2026	

7. Rúbrica de Calificación

7.1 Requisitos para optar a la calificación

Antes de la evaluación de la práctica, los estudiantes deben cumplir con los requisitos que se indiquen en esta sección.

Software – Introducción a la Programación 1

Tema	**Descripción**	**Cumple (Sí/No)**
Cumplimiento de la tecnología establecida	El desarrollo debe haberse realizado en **Java**, ser una aplicación de **Consola** y usar **vectores (arreglos) nativos**.	
Uso de herramientas requeridas	El proyecto debe compilar y ejecutarse en NetBeans, IntelliJ o similar.	
Gestión y entregas de práctica	El repositorio en **GitHub** debe ser privado, tener el nombre `IPC1_Practica1_#Carné` y tener al auxiliar como colaborador.	
Documentación obligatoria	Se debe incluir Manual Técnico, Manual de Usuario y Diagrama de Flujo en PDF.	

Pruebas y funcionalidad mínima	El sistema debe, como mínimo, mostrar el Menú de Inicio, permitir Iniciar una partida (en cualquier tablero), y que el personaje Pac-Man se mueva en el tablero.	

7.2 Resumen de Puntuaciones

Software – Introducción a la Programación 1

Área	**Puntos Totales**	**Puntos Obtenidos**
1. Habilidades (Cómo se programa)		
Uso de Vectores (Arreglos nativos)	**15**	
Calidad de código (modularidad, legibilidad)	**15**	
Entrega & Repositorio (Formato, .jar ejecutable)	**10**	
Sub-Total Habilidades	**40**	
2. Conocimiento (Qué se programa)		
Menú de Inicio e Historial de Partidas	**15**	
Configuración de Partida (Validaciones %, Aleatoriedad)	**15**	

Dinámica del Juego (Movimiento, Colisiones, Puntos, Vidas)	**25**	
Lógica de Pausa y Fin de Partida (Gana/Pierde)	**5**	
Sub-Total Conocimiento	**60**	
TOTAL	**100**	

Detalle de la Calificación

No.	Criterio de evaluación	Punteo maximo	Nivel de evaluación		Punteo Obtenido
			Satisfactorio (100% - 61%)	**Necesita mejorar** (60% - 0%)	
1	**Habilidades**	**40**			

1.1	**Uso de Vectores (Arreglos nativos)**	15	El programa utiliza arreglos nativos (vectores) de forma eficiente para gestionar el tablero, la posición de los ítems y el historial de partidas. Cumple estrictamente con la restricción de NO usar ArrayLists, LinkedLists u otras librerías de colecciones.	El programa utiliza librerías no permitidas (ArrayList, etc.) o maneja los arreglos nativos de forma incorrecta, causando errores de lógica (ArrayIndexOutOfBounds) o no resolviendo los problemas planteados.	
1.2	**Calidad de código (modularidad, legibilidad)**	15	El código es modular (divide la lógica en métodos/funciones), es legible, está bien comentado y sigue buenas prácticas. Evita la repetición innecesaria de código (DRY).	El código está escrito mayormente en el 'main', es difícil de leer, no está comentado y repite lógica de forma excesiva, dificultando su mantenimiento.	

1.3	**Entrega & Repositorio (Formato, .jar)**	10	El repositorio (nombre, privacidad, colaborador) cumple al 100% con lo solicitado. El archivo .jar se incluye y es ejecutable, permitiendo jugar la partida sin necesidad de compilar el código.	El repositorio tiene el nombre incorrecto, es público, falta el colaborador o faltan archivos. El .jar no se incluye, está corrupto o no se ejecuta.	
	Sub-Total de Puntos				
2	**Conocimientos**	**60**			

2.1	**Menú de Inicio e Historial de Partidas**	15	El Menú de Inicio es 100% funcional (Iniciar, Historial, Salir). El historial de partidas se muestra correctamente y está ordenado del más reciente al más antiguo, como fue solicitado.	El menú principal tiene opciones que no funcionan (especialmente Salir). El historial no se muestra, se muestra vacío o no respeta el orden cronológico inverso.	

2.2	**Configuración de Partida (Validaciones %, Aleatoriedad)**	15	El sistema solicita el nombre, tipo de tablero y las cantidades. Valida correctamente que las cantidades de ítems no superen los porcentajes máximos (40%, 20%, 20%) y sean al menos 1. Los ítems se posicionan aleatoriamente.	El sistema no valida las entradas del usuario (permite valores negativos o superiores al límite), causando errores. La aleatoriedad no se aplica (los ítems siempre salen en el mismo lugar).	

2.3	**Dinámica del Juego (Movimiento, Colisiones, Puntos, Vidas)**	25	Pac-Man se mueve correctamente con las teclas (8,5,4,6). Las colisiones son perfectas: se detiene ante paredes (X), pierde vida con fantasmas (@) y suma puntos con premios (0, $). La lógica de "bordes infinitos" (reaparecer al otro lado) funciona.	El personaje no se mueve o lo hace con otras teclas. El personaje atraviesa paredes, no suma puntos al comer premios o no pierde vidas con los fantasmas. La lógica de bordes falla.	

2.4	**Lógica de Pausa y Fin de Partida (Gana/Pierde)**	5	El menú de pausa (tecla 'F') funciona y sus opciones (Regresar, Terminar) se ejecutan correctamente. El juego detecta automáticamente cuándo el jugador gana (obtiene todos los premios) o pierde (llega a 0 vidas) y termina la partida, registrándola en el historial.	La tecla de pausa no funciona, o las opciones del menú no responden. El juego continúa indefinidamente aunque el jugador haya perdido todas las vidas o comido todos los premios.	
	Sub-Total de Puntos				
	Total	**100**			