



UNIVERSIDAD ESTATAL A DISTANCIA
VICERRECTORÍA ACADÉMICA
ESCUELA DE CIENCIAS EXACTAS Y NATURALES



Cátedra Tecnología de Sistemas

Programación Intermedia

Código: 00824

Proyecto 1. Valor 1%

Entrega: Semana 3

1. Temas de Estudio

Lista de los temas/unidades del curso que se aplican en el proyecto.

- Introducción a las computadoras, Internet y Java.
- Aplicaciones en Java: entrada/salida y operadores.
- Instrucciones de control (selectivas, repetitivas).
- Métodos.
- Arreglos y objetos ArrayList.
- Clases y objetos: introducción y análisis detallado.

2. Objetivo

Aplicar estructuras de control, arreglos, colecciones y clases en Java mediante el desarrollo de una aplicación en modo consola que gestione el servicio de parqueo de automóviles, fortaleciendo las habilidades de programación orientada a objetos y resolución de problemas.

3. Software de Desarrollo

- Nombre del software de desarrollo: Java
- Entorno de programación oficial: NetBeans IDE
- Paradigma: Programación orientada a objetos

4. Desarrollo: Requerimientos técnicos y funcionales

Una empresa de servicios urbanos ha decidido implementar un sistema automatizado para la gestión de parqueos en sus instalaciones. El sistema debe permitir registrar el ingreso y salida de vehículos, administrar los espacios disponibles, calcular las tarifas del servicio de parqueo (por hora o por día) según el tipo de servicios y de vehículo, y generar reportes básicos que apoyen la toma de decisiones a nivel operativo y financiero. El desarrollo se realizará en Java, **en modo consola**, aplicando los principios de la programación orientada a objetos (clases, arreglos y colecciones). La aplicación debe ejecutarse a partir de la clase principal (Main) y ofrecer un menú de opciones que facilite el acceso a las distintas funcionalidades del sistema.

El sistema debe contemplar las siguientes funcionalidades específicas:

1. Registro de vehículos:

- Datos obligatorios: placa, El sistema debe solicitar el **tipo de vehículo** mediante un valor numérico, con las siguientes opciones (1 = Automóvil, 2 = Motocicleta, 3 = Camión), hora y fecha de entrada al parqueo.
- La fecha y hora de ingreso deben ser tomadas automáticamente del sistema y almacenadas junto con el resto de los datos del vehículo (**no debe ser digitadas manualmente por la persona usuaria**).
- Validación de formato y unicidad de la placa:
 1. **Formato:** La placa debe cumplir con el patrón de **tres letras seguidas de tres números** (por ejemplo: ABC123). No se permiten placas compuestas únicamente por números.

2. **Unicidad:** El sistema no debe permitir el registro de un vehículo cuya placa ya esté registrada como “**dentro del parqueo**”, evitando duplicidad en el control de ingreso.
3. En caso de que la placa no cumpla el formato establecido o ya exista un vehículo activo con esa placa, el sistema debe mostrar un mensaje de error adecuado y no registrar la información.

2. Cálculo de tarifas:

Las tarifas del servicio del parqueo deben calcularse siguiendo las siguientes reglas:

- **Tarifas diferenciadas por tipo de vehículo:** El sistema debe aplicar tarifas específicas según el tipo de vehículo. Las motocicletas pagan ¢500 por hora y ¢3000 por día, mientras que los automóviles y camiones pagan ¢600 por hora y ¢5000 por día.
- **Selección del tipo de servicio (por hora y por día):** Al registrar el vehículo, el usuario debe seleccionar el tipo de servicio que desea contratar (por hora o por día). El sistema utilizará esta información para calcular el monto total a pagar según el tiempo de permanencia.
- **Cálculo automático del monto a pagar según tiempo de permanencia:** El sistema debe calcular de forma automática el total a pagar tomando en cuenta: el tipo de vehículo, el tipo de servicio (por hora o por día) y el tiempo transcurrido entre la hora de entrada y la hora de salida.
- **Aplicación de descuentos por tiempo prolongado:** Si el servicio es por hora y el vehículo permanece más de 8 horas dentro del parqueo, se debe aplicar un descuento del 10% sobre el total calculado. Este beneficio no aplica para el servicio por día.

3. Salida de vehículos:

- **Registro de hora de salida:** El sistema debe permitir ingresar la hora de salida del vehículo, validando que sea posterior a la hora de entrada y que cumpla con el

formato establecido. Esta información es indispensable para calcular el tiempo de permanencia en el parqueo.

- **Cálculo del total a pagar:** Utilizando estructuras de control y operadores aritméticos, el programa debe calcular automáticamente el monto total a pagar, considerando: el tipo de vehículo, el tipo de servicio (por hora o por día), el tiempo de uso del parqueo y la aplicación de descuentos cuando corresponda, según las reglas definidas en el apartado de cálculo de tarifas
- **Generación de recibo en consola:** Al finalizar el servicio, el sistema debe mostrar en la consola un recibo detallado que incluya, como mínimo: placa del vehículo, tipo de vehículo, tipo de servicio, tiempo de permanencia, tarifa aplicada, descuento (si corresponde) y total a pagar. Este recibo debe estar bien estructurado y ser fácilmente legible para el usuario.

4. Gestión de parqueo:

- **Capacidad máxima del parqueo:** El sistema debe considerar que el parqueo cuenta con una capacidad limitada de 10 espacios. Esta cantidad representa el número máximo de vehículos que pueden estar registrados simultáneamente como "dentro del parqueo" en el sistema.
- **Control de espacios disponibles:** El sistema debe implementar una estructura de datos, como un arreglo o un ArrayList, para gestionar los vehículos que actualmente ocupan un espacio en el parqueo. Esta estructura debe almacenar información esencial de cada vehículo. Cada vez que un vehículo ingrese o salga del parqueo, el sistema debe actualizar esta estructura para reflejar el estado actual de ocupación.
- **Mensajes de advertencia por nivel de ocupación:** El sistema debe mostrar un mensaje de advertencia cuando el parqueo alcance el 50% de ocupación o más (es decir, 5 vehículos o más). Si se alcanza la capacidad máxima (10 vehículos), el sistema debe impedir nuevos ingresos y mostrar un mensaje indicando que el parqueo está completo.

5. Reportes:

- **Listado de vehículos atendidos en el día:** El sistema debe generar en consola un listado en consola con todos los vehículos que han utilizado el parqueo durante la jornada actual, mostrando como mínimo los siguientes datos: placa, tipo de vehículo, tipo de servicio, tiempo de permanencia y monto pagado.
- **Total recaudado por tipo de servicio:** El programa debe calcular y mostrar el total de ingresos obtenidos por cada tipo de servicio (por hora y por día), permitiendo al administrador conocer el rendimiento económico del parqueo.
- **Estadísticas básicas de uso del parqueo:** El sistema debe incorporar indicadores esenciales, como la **sumatoria de horas de uso**, la **cantidad de vehículos atendidos** y el **porcentaje de ocupación alcanzado durante la jornada**. Estas métricas permitirán evaluar la eficiencia del servicio y respaldar la toma de decisiones orientadas a la mejora continua.

6. Estructuras de control:

- Uso de if, switch, for, while, do-while para controlar el flujo del programa.

7. Estructuras de datos:

- Uso de ArrayList para almacenar los registros de vehículos y servicios.
- Uso de arreglos para estadísticas simples.

8. Clases y métodos:

- **Clases sugeridas:**

El proyecto debe estructurarse utilizando clases que representen los componentes principales del sistema. Se recomienda implementar al menos las siguientes:

1. Vehículo: para almacenar los datos del vehículo (placa, tipo, hora de entrada y salida).
2. ServicioParqueo: para gestionar el tipo de servicio (por hora o por día), tarifas y cálculos asociados.
3. GestorParqueo: para controlar la ocupación del parqueo, validar espacios disponibles y administrar el flujo de vehículos.
4. Reporte: para generar listados, totales recaudados y estadísticas de uso.

Métodos funcionales del sistema: Cada clase debe implementar métodos con las siguientes funciones principales:

- **Registrar información:** Permitir el ingreso y almacenamiento de los datos requeridos por el sistema.
- **Calcular tarifas y descuentos:** Realizar los cálculos necesarios según las reglas definidas en el proyecto.
- **Validar entradas y condiciones:** Comprobar que los datos ingresados cumplen con el formato y las restricciones establecidas.
- **Mostrar resultados en consola:** Presentar la información de manera clara, organizada y legible para el usuario.

Estos métodos deben aplicar correctamente los conceptos de programación orientada a objetos, estructuras de control y manejo de datos.

9. Manejo de excepciones:

- El sistema debe contemplar el manejo de errores frecuentes mediante el uso adecuado de bloques try-catch y validades de entrada. Entre otros, se deben considerar casos como: entradas vacías, formatos incorrectos, división por cero.
- Mensajes claros y amigables para el usuario: Ante cualquier error o dato inválido, el programa debe mostrar mensajes claros y amigables en la consola, que indiquen al usuario que ocurrió y que acción debe realizar (por ejemplo: volver a digitar un dato).

10. Documentación interna:

- **Comentarios explicativos en el código:** El código fuente debe incluir comentarios internos que expliquen, de manera breve y pertinente, la finalidad de las clases, métodos y secciones principales del programa. No se deben comentar línea por línea, sino aquellos fragmentos que requieren aclaración para su lectura y mantenimiento.
- **Declaración de autoría y fuentes externas utilizadas:** El proyecto debe incluir una sección de encabezado en el código, donde se indique el nombre completo del estudiante, grupo, curso, cuatrimestre, así como la mención de cualquier material o recurso externo utilizado (libros, sitios WEB, IAG, entre otros).

Ejemplos de entradas y salidas esperadas:**1. Menú Principal – Sistema de Parqueo:**

Al ejecutar la aplicación, se debe mostrar un menú similar al siguiente:

```
=====
                SISTEMA DE ADMINISTRACIÓN DE PARQUEO
=====

Fecha actual: 21/10/2025

Seleccione una opción:
1. Registrar ingreso de vehículo
2. Registrar salida de vehículo
3. Consultar espacios disponibles
4. Ver reportes del día
5. Salir del sistema

Ingrese su opción: _
```

2. Menú de Reportes:

Cuando el usuario seleccione la opción de reportes, el sistema debe mostrar un menú similar al siguiente:

```
=====
                MENÚ DE REPORTES
=====

Fecha actual: 21/10/2025

Seleccione el reporte que desea consultar:
1. Listado de vehículos atendidos
2. Total recaudado por tipo de servicio
3. Estadísticas básicas de uso del parqueo
4. Volver al menú principal

Ingrese su opción: _
```


3. Ingreso de datos:

Ejemplo de interacción al registrar el ingreso de un vehículo:

```
Ingrese la placa del vehículo (formato ABC123): ABC456
Seleccione el tipo de vehículo:
1. Automóvil
2. Motocicleta
3. Camión
Opción: 1
Ingrese la hora de entrada (formato 24 horas, solo hora): 08
```

4. Datos del vehículo:

Ejemplo de cómo podría mostrarse en consola la información registrada para un vehículo:

```
Placa: ABC456
Tipo de vehículo: Automóvil
Tipo de servicio: Por hora
Hora de entrada: 08
Hora de salida: 17
Fecha: 21/10/2025
```

5. Cálculo del tiempo de permanencia:

A partir de la hora de entrada y la hora de salida, el sistema calcula el tiempo de uso y el monto a pagar, un ejemplo de cómo podría mostrarse esta información en consola es el siguiente:

```
Horas de uso: 9
Tarifa por hora (automóvil): $600
Subtotal: $5400
Descuento aplicado (10% por más de 8 horas): $540
Total a pagar: $4860
```

6. Recibo generado en consola:

Ejemplo de recibo mostrado en consola al finalizar el servicio de parqueo:

```
-----
                        RECIBO DE SERVICIO DE PARQUEO
-----
Placa del vehículo: ABC456
Tipo de vehículo: Automóvil
Tipo de servicio: Por hora
Fecha de ingreso: 21/10/2025
Hora de entrada: 08:00
Hora de salida: 17:00
Horas de uso: 9
Tarifa por hora: $600
Subtotal: $5400
Descuento aplicado (10%): $540
-----
TOTAL A PAGAR: $4860
-----
Gracias por utilizar nuestro servicio.
```

Salida esperada – Servicio por día (sin descuento)

7. Datos del vehículo:

Ejemplo de cómo podría mostrarse en consola la información registrada para un vehículo con servicio por día:

```
Placa: MOT0123  
Tipo de vehículo: Motocicleta  
Tipo de servicio: Por día  
Fecha de ingreso: 21/10/2025
```

8. Cálculo del monto:

Ejemplo de cálculo del monto a pagar para una motocicleta con servicio por día:

```
Tarifa diaria (motocicleta): $3000  
Total a pagar: $3000
```

9. Recibo generado en consola:

Ejemplo de recibo mostrado en consola para una motocicleta con servicio por día:

```
-----  
      RECIBO DE SERVICIO DE PARQUEO  
-----  
Placa del vehículo:ABC123  
Tipo de vehículo: Motocicleta  
Tipo de servicio: Por día  
Fecha de ingreso: 21/10/2025  
-----  
TOTAL A PAGAR: $3000  
-----  
Gracias por utilizar nuestro serv-  
icio.  
Gracias por utilizar nuestro  
servicio.  
-----
```

10. Salida esperada – Control de ocupación del parqueo:

Ejemplo de mensaje mostrado en consola cuando se registre un vehículo y el parqueo alcanza el 50% de ocupación:

```
Vehículo registrado correctamente.  
Espacios ocupados: 5 / 10  
⚠ Advertencia: El parqueo ha alcanzado el 50% de su capacidad.
```

11. Salida en consola al registrar el décimo vehículo:

Ejemplo de mensaje mostrado en consola cuando se registra el décimo vehículo y el parqueo llega a su capacidad máxima:

```
Vehículo registrado correctamente.  
Espacios ocupados: 10 / 10  
🚫 El parqueo está completo. No se permiten más ingresos.
```

12. Un vehículo sale del parqueo, liberando un espacio.

Ejemplo de mensaje mostrado en consola cuando un vehículo abandona el parqueo y se libera un espacio:

```
Vehículo con placa ABC123 ha salido del parqueo.  
Espacios ocupados: 9 / 10  
✅ Espacio disponible. Puede ingresar un nuevo vehículo.
```

13. Listado de vehículos atendidos en el día

Ejemplo de reporte mostrado en consola con los vehículos atendidos durante la jornada:

```
=====
LISTADO DE VEHÍCULOS ATENDIDOS
Fecha: 21/10/2025
=====

1. Placa: ABC123 | Tipo: Automóvil | Servicio: Por hora | Tiempo: 4 horas | Total
pagado: $2400
2. Placa: MOTO456 | Tipo: Motocicleta | Servicio: Por día | Tiempo: 1 día | Total
pagado: $3000
3. Placa: CAM789 | Tipo: Camión | Servicio: Por hora | Tiempo: 9 horas | Total pagado:
$4860
4. Placa: XYZ321 | Tipo: Automóvil | Servicio: Por día | Tiempo: 1 día | Total pagado:
$5000
5. Placa: MOTO999 | Tipo: Motocicleta | Servicio: Por hora | Tiempo: 2 horas | Total
pagado: $1000

Total de vehículos atendidos: 5
=====
```

14. Total recaudado por tipo de servicio

Ejemplo de reporte mostrado en consola con los ingresos según el tipo de servicio:

```
=====
                INGRESOS POR TIPO DE SERVICIO
                Fecha: 21/10/2025
=====

♦ Servicio por hora:
- Total recaudado: $8260

♦ Servicio por día:
- Total recaudado: $8000

📊 Total general del día: $16,260
=====
```

15. Estadísticas básicas de uso del parqueo

Ejemplo de reporte mostrado en consola con las estadísticas de uso del parqueo:

```
=====
                ESTADÍSTICAS DE USO DEL PARQUEO
                Fecha: 21/10/2025
=====

♦ Promedio de tiempo de permanencia: 4.2 horas
♦ Porcentaje de ocupación alcanzado: 50%
♦ Vehículos atendidos: 5
♦ Espacios disponibles al cierre: 5 / 10

🚧 Observación: El parqueo alcanzó el 50% de ocupación durante la jornada.
=====
```



<https://audiovisuales.uned.ac.cr/play/player/23048>

1. Entregables

- Es obligatorio que incluya todo el directorio donde se encuentra el Proyecto # 1.
- El Proyecto #1 debe desarrollarse en el lenguaje de programación Java en el entorno de programación NetBeans IDE que es la herramienta oficial del curso.
- Los trabajos deben realizarse en forma individual; no se permiten trabajos grupales.
- Dentro del código del programa debe de indicar la documentación interna (comentarios) que explique cómo está organizado y cómo funciona el programa.
- Si utiliza código de algún ejemplo del libro, del algún profesor o de otra fuente que no sea de su autoría, debe de indicarlo.
- Comprima todos los archivos en un solo archivo .zip o .rar.
- **Nombre del archivo que envía:** debe ser nombre y primer apellido del estudiante, y nombre de la tarea. **Ejemplo: JuanRojas-Proyecto1.**
- La entrega del proyecto debe ser en las fechas establecidas en la plataforma de aprendizaje en línea Moodle, en el apartado que se indique.
- Si no concluyó a tiempo el proyecto, debe entregar lo que pudo hacer e incluir una carta explicando las razones por las cuales no finalizó.

Nota importante

- Cada estudiante es responsable del contenido que entrega.
- Si el archivo enviado no corresponde al proyecto correcto, no se aceptarán entregas posteriores fuera de la fecha establecida.
- Si se detecta que el contenido del archivo coincide con otro estudiante o no es de su autoría, se aplicarán las sanciones indicadas en el documento de Lineamientos ante casos de plagio disponible en la plataforma.
- En caso de utilizar herramientas de Inteligencia Artificial Generativa para apoyar el desarrollo del proyecto el estudiante debe:
 - Indicarlo explícitamente en la documentación del proyecto (comentarios en el código y/o una breve sección en el documento o carta adjunta).
 - Revisar, adaptar y comprender todo el código generado.
 - Asumir la responsabilidad total sobre el funcionamiento del sistema.
 - Indicar cuáles herramientas utilizó.
 - Adjuntar a la documentación los Prompt(s) principales que utilizó y cuyo resultado fue incorporado, total o parcial, en el proyecto.
 - El uso de IA generativa para entregar un sistema que el estudiante no comprende o no pueda explicar podrá considerarse una falta a la honestidad académica y, según la gravedad, tratarse como plagio, de acuerdo con los Lineamientos ante casos de plagio y la normativa vigente de la UNED.



PROYECTOS PROGRAMADOS

ANTES DE ENVIAR EL PROYECTO

1. Asegúrese de que el programa **compile** y se ejecute correctamente.
2. Revise el archivo comprimido (.zip) antes de enviarlo para confirmar que incluye **todos los archivos necesarios** y que el proyecto funciona sin errores.
3. Si el proyecto **compila pero no se ejecuta correctamente**, o si **no compila**, **se evaluará según los criterios establecidos en la rúbrica**.
4. Si la calificación final, según la rúbrica, es 0 (cero), **no podrá realizar la defensa oral del proyecto**.
5. Si no entrega el proyecto (archivo .zip), su calificación será **0 (cero)**, salvo en **casos de fuerza mayor debidamente justificados** y tramitados conforme al Reglamento General Estudiantil.

2. Rúbrica de Evaluación

Criterio	Excelente	Bueno	Aceptable	Insuficiente
1. Uso de estructuras de control	Aplica correctamente if, switch, for, while, do-while en coherencia con el problema planteado. (15 puntos)	Aplica la mayoría de las estructuras de control requeridas, con leves omisiones o errores que no afectan el funcionamiento o general del programa. (11 puntos)	Aplica solo algunas estructuras de control o presenta errores que afectan casos específicos del programa. (7 puntos)	No utiliza estructuras de control o las usa de forma incorrecta, impidiendo el correcto funcionamiento o del programa. (0 puntos)
2. Diseño de clases y métodos (15 puntos)	Implementa correctamente las clases sugeridas (Vehiculo, ServicioParqueo, GestorParqueo, Reporte) con métodos funcionales, buena	Implementa la mayoría de las clases sugeridas con una estructura adecuada y métodos funcionales, aunque con algunos detalles de diseño que pueden ser	Implementa parcialmente las clases o método, o presenta errores de diseño que afectan el uso de la solución. (7 puntos)	No aplica POO o lo hace de forma incorrecta (clases incompletas o sin uso adecuado). (0 puntos)

Criterio	Excelente	Bueno	Aceptable	Insuficiente
	organización interna. (15 puntos)	mejorados. (11 puntos)		
3. Algoritmo y lógica de cálculo (10 puntos)	El algoritmo claro, coherente y resuelve correctamente el cálculo de tarifas, descuentos y ocupación del parqueo. (10 puntos)	El algoritmo es adecuado y resuelve los cálculos principales, con fallos menores en algunos casos. (7 puntos)	El algoritmo es incompleto o presenta errores que afectan el resultado en varios escenarios. (5 puntos)	No resuelve el problema o no presenta algoritmo válido para los cálculos. (0 puntos)
4. Entrada y salida de datos (10 puntos)	Implementa correctamente la Entrada de datos con validaciones (placa, tipo de vehículo, hora) y salidas claras (recibo, reportes) y genera salidas claras	Entradas/salidas as mayormente correctas con algunos detalles menores de validación o presentación. (7 puntos)	Entradas/salidas limitadas o poco claras, faltan datos relevantes o hay mensajes confusos. (5 puntos)	No implementa adecuadamente la entrada/salida o de datos o esta impide el uso correcto del sistema. (0 puntos)

Criterio	Excelente	Bueno	Aceptable	Insuficiente
	y completas (recibos, mensajes y reportes). (10 puntos)			
5. Uso de operadores y estructuras básicas de Java (10 puntos)	Emplea correctamente operadores aritméticos/lógicos, así como estructuras como ArrayList de acuerdo con los requerimientos del proyecto. (10 puntos)	Utiliza operadores y estructuras de datos con algunos fallos menores o un uso parcial de las mismas. (7 puntos)	Uso limitado de operadores y/o estructuras de datos, con errores frecuentes que afectan el resultado. (5 puntos)	No utiliza adecuadamente los operadores o estructuras de datos solicitadas. (0 puntos)
6. Generación de reportes (10 puntos)	Genera correctamente los tres reportes: listado de vehículos atendidos, ingresos por	Genera correctamente dos de los reportes requeridos. (7 puntos)	Genera solo uno de los reportes o presenta errores significativos en los resultados.	No genera reportes o los reportes son incompletos e inutilizables. (0 puntos)

Criterio	Excelente	Bueno	Aceptable	Insuficiente
	tipo de servicio y estadísticas básicas de uso del parqueo. (10 puntos)		(5 puntos)	
7. Menú interactivo funcional (10 puntos)	Implementa un menú claro, funcional y fácil de navegar, incluyendo el acceso a las principales opciones del sistema (registro, salida, consulta de espacios, reportes y salida del programa). (10 puntos)	Menú funcional con detalles menores de navegación o presentación. (7 puntos)	Menú limitado o con errores de navegación que dificultan el uso del sistema. (5 puntos)	No implementa un menú interactivo o este resulta inoperable. (0 puntos)

Criterio	Excelente	Bueno	Aceptable	Insuficiente
8. Manejo de excepciones (10 puntos)	Maneja adecuadamente excepciones y errores comunes (formato, división por cero) mostrando mensajes claros. (10 puntos)	Maneja algunos errores y excepciones con mensajes adecuados, aunque no cubre todos los casos relevantes. (7 puntos)	Manejo limitado de errores, los mensajes son poco claros o no guían adecuadamente al usuario. (5 puntos)	No implementa manejo de excepciones o el programa se detiene ante errores de entrada. (0 puntos)
9. Estructura ejecutable desde clase Main (5 puntos)	El programa inicia correctamente desde una clase Main bien estructurada. (5 puntos)	El programa inicia desde Main con algunos detalles menores de organización. (4 puntos)	El programa inicia desde Main pero la estructura es confusa o poca organizada (3 puntos)	No inicia desde la clase Main o no se puede ejecutar el proyecto. (0 puntos)
10. Documentación y entrega (5 puntos)	Código bien comentado, indentado, entregado en formato correcto	Comentarios parciales y formato de entrega adecuado, con pequeños	Documentación limitada (pocos comentarios, indentación deficiente) o	Sin documentación interna o entrega en un formato incorrecto

Criterio	Excelente	Bueno	Aceptable	Insuficiente
	(.zip/.rar), con nombre de archivo según la indicación, e incluye declaración de autoría y fuentes externa utilizadas. (5 puntos)	detalles por corregir. (4 puntos)	errores en el formato de entrega (nombre del archivo, compresión, etc) o en la ortografía. (3 puntos)	/incompleto que dificulta la revisión. (0 puntos)

Total: 100 puntos