



Guía para el profesor

Programación orientada a objetos
LTTI2112



Índice

Información general del curso.....	1
Metodología.....	2
Evaluación.....	4
Bibliografía.....	5
Tips importantes.....	6
Temario.....	7
Notas de enseñanza.....	7
Evidencia final.....	10

Información general del curso

Modalidades

- Clave banner: LTTI2112
- Modalidad: Presencial, en línea y Connect

Competencia del curso

Soluciona problemas computacionales mediante el uso del paradigma de programación orientada a objetos.





Metodología

Metodología Connect

Se ha diseñado un curso **Connect** con la finalidad de ser impartido por un **docente líder con experiencia en el ámbito laboral**, quien compartirá contigo su conocimiento, experiencia y las mejores prácticas que realiza en su labor profesional.

La experiencia de cursos Connect promueve la interacción virtual entre estudiantes localizados en diferentes campus de la Universidad Tecmilenio, como una forma de enriquecer tu formación, contrastando la realidad de tu ciudad o región con la de otros compañeros.

Durante cada sesión virtual, el docente transmite su experiencia y actúa como guía en el proceso de aprendizaje durante la realización de las actividades.

El curso es **tetramestral** y tiene una distribución **semanal**; en cada semana se lleva a cabo una sesión virtual sincrónica de tres horas a través de una herramienta tecnológica de videoconferencia. La asistencia a estas sesiones de videoconferencia es muy importante, pero no obligatoria, ya que tienes la posibilidad de revisar la sesión grabada en caso de no poder asistir en el horario establecido.

Estructura de las sesiones

Las sesiones se dividen en tres bloques; estas son las actividades que se recomienda realizar:

Bloque 1	Bloque 2	Bloque 3
<ul style="list-style-type: none"> • Bienvenida y presentación de agenda. • Actividad de bienestar-mindfulness. • Desarrollo de temas de la semana. <ul style="list-style-type: none"> ○ Aplicación en contextos reales. ○ Actividades. 	<ul style="list-style-type: none"> • Actividad de reconexión. • Desarrollo de temas de la semana. <ul style="list-style-type: none"> ○ Aplicación en contextos reales. ○ Actividades. ○ Cierre del tema. • Cierre del bloque mediante utilización 	<ul style="list-style-type: none"> • Actividad de reconexión. • Desarrollo de temas de la semana. <ul style="list-style-type: none"> ○ Aplicación en contextos reales. ○ Actividades. ○ Cierre del tema. • Cierre de la sesión mediante utilización

Bloque 1	Bloque 2	Bloque 3
<ul style="list-style-type: none"> ○ Cierre del tema. • Cierre del bloque mediante utilización de herramientas tecnológicas diversas para una dinámica, foro o quiz. • Receso. 	<ul style="list-style-type: none"> de herramientas tecnológicas diversas para una dinámica, foro o quiz. • Receso. 	<ul style="list-style-type: none"> de herramientas tecnológicas diversas para una dinámica, foro o quiz.

Metodología Online

Este curso ha sido diseñado para desarrollar la competencia expresa en el apartado correspondiente. La estructura del curso consiste en 8 temas dividido en dos módulos.

Cada tema incluirá material didáctico compuesto por lecturas, videos y otros recursos de apoyo para facilitar la comprensión. Los alumnos deberán dedicar al menos 10 horas semanales para la revisión de los temas, actividades, evidencias y la resolución de exámenes. Se espera que los alumnos gestionen su tiempo de estudio de manera efectiva y desarrollen la autodisciplina necesaria para cumplir con las horas de estudio recomendadas.

Los alumnos realizarán actividades relacionadas con cada tema para aplicar los conocimientos adquiridos. Se evaluará a los estudiantes mediante un examen final que abarcará los dos módulos y medirá la comprensión global de la competencia.

La comunicación efectiva será clave. Se utilizarán plataformas en línea y herramientas de mensajería para mantener a los alumnos informados sobre cualquier actualización o recordatorios importantes. Los alumnos deben aclarar sus dudas a través de canales establecidos, como foros en línea, correos electrónicos o sesiones de consulta programadas.

Esta metodología busca garantizar un aprendizaje profundo y significativo, promoviendo la participación del estudiante y su compromiso con el desarrollo de la competencia establecida.

Metodología Presencial

Este curso ha sido diseñado para desarrollar la competencia expresa en el apartado correspondiente. La estructura del curso consiste en 8 temas dividido en dos módulos.

Cada tema incluirá material didáctico compuesto por lecturas, videos y otros recursos de apoyo para facilitar la comprensión. Los alumnos deberán dedicar al menos 10 horas semanales para la revisión de los temas, actividades, evidencias y la resolución de exámenes. Se espera que los alumnos gestionen su tiempo de estudio de manera efectiva y desarrollen la autodisciplina necesaria para cumplir con las horas de estudio recomendadas.

Los alumnos realizarán actividades relacionadas con cada tema para aplicar los conocimientos adquiridos. Se evaluará a los estudiantes mediante un examen final que abarcará los dos módulos y medirá la comprensión global de la competencia.

La comunicación efectiva será clave. Se utilizarán plataformas en línea y herramientas de mensajería para mantener a los alumnos informados sobre cualquier actualización o recordatorios importantes. Los alumnos deben aclarar sus dudas a través de canales establecidos, como foros en línea, correos electrónicos o sesiones de consulta programadas.

Esta metodología busca garantizar un aprendizaje profundo y significativo, promoviendo la participación del estudiante y su compromiso con el desarrollo de la competencia establecida.



Evaluación

Número	Evaluable	Ponderación
1	Examen 1	20
2	Evidencia 1	30
3	Examen 2	20
4	Evidencia 2	30
Total		100



Bibliografía

➔ Libro de texto

Joyanes, L., y Zahonero, I. (2014). *Programación en C, C++, Java Y UML* (2ª ed.). México: McGraw Hill. ISBN: 9786071512123

➔ Libros de apoyo

Joyanes, L. (2013). *Programación en Java 6*. México: McGraw Hill. ISBN eBook: 9781456205584

Malik, D. (2013). *Programación JAVA del Análisis de Problemas al Diseño de programas* (6ª ed.). (eBook). México: Cengage Learning. ISBN-10: 6074819262, ISBN-13: 9786074819267

➔ Requerimientos especiales

Netbeans. Disponible en <https://netbeans.org/downloads/>

JDK (Java Development Kit). Disponible en <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

starUML. Disponibles en <http://staruml.io/>

BlueJ. Disponible en <http://www.bluej.org/index.html>





Tips importantes

1. Material de capacitación en la plataforma tecnológica Canvas

- Tutorial digital para profesores: <https://bit.ly/2SbMaNK>
- Tutorial digital para alumnos: <https://bit.ly/35lBnP6>
- Para revisar los videotutoriales de la Biblioteca Digital, da clic [aquí](#).

2. ¿En dónde o a quién reporto un error detectado en el contenido del curso?

Lo puedes reportar a la cuenta atencioncursos@servicios.tecmilenio.mx, pero te pedimos que también reportes sugerencias para el contenido y actividades del curso.

3. ¿Quién me informa de la cantidad de sesiones y tiempo de cada sesión en las semanas?

El coordinador docente te debe proporcionar esta información.

4. ¿En qué semanas se aplican los exámenes parciales y el examen final?

Consulta con tu coordinador docente los calendarios de acuerdo con la modalidad de impartición.

5. ¿Tengo que capturar las calificaciones en banner y en la plataforma educativa?

Sí, es importante que captures calificaciones en la plataforma para que los alumnos estén informados de su avance y reciban retroalimentación de parte tuya de todo lo que realizan en el curso. En banner es el registro oficial de las calificaciones de los alumnos.

Las actividades evaluables dentro de cada tema y del calendario, así como la presentación final del curso tienen asignadas una rúbrica con la cual es obligatorio que se califique, esto es

muy importante para nuestro modelo de competencias, ya que es la forma en la que medimos el desarrollo de las competencias en nuestros alumnos. Cada rúbrica se encuentra en la sección de **tareas**, específicamente con el nombre de la semana evaluable.

Es importante evaluar con la rúbrica que aparece en el apartado de la evidencia final, ya que se les estará auditando constantemente su realización efectiva.

Video disponible para calificar con rúbricas.

- ¿Cómo busco una rúbrica?: <https://youtu.be/QgDKeZvgtAI>
- ¿Cómo califico con una rúbrica?: <https://youtu.be/mAblsLAGlp4>



Temario

1	Conceptos fundamentales de programación orientada a objetos
2	Modelado de clases
3	Introducción al lenguaje orientada a objetos
4	Estructura de repetición
5	Métodos y parámetros
6	Arreglos
7	Cadenas de caracteres
8	Manejo de excepciones
9	Clases
10	Constructores
11	Otros conceptos de clases y objetos
12	Objetos que contienen objetos
13	Colecciones
14	Herencia
15	Poliformismo



Notas de enseñanza

Tema 1 Conceptos fundamentales de programación orientada a objetos

Usa analogías con el mundo real para explicar objetos y clases. Incluye ejemplos visuales para mostrar la interacción entre objetos. Organiza una lluvia de ideas con ejemplos cotidianos.

Tema 2 Modelado de clases

Utiliza diagramas UML para enseñar las relaciones entre clases. Propón actividades prácticas con problemas cotidianos. Facilita herramientas de diseño visual para explorar modelos.

Tema 3 Introducción al lenguaje orientado a objetos

Inicia con código sencillo y comentado. Proporciona fragmentos de código incompletos para que los estudiantes los completen. Ofrece retroalimentación rápida mediante pequeños proyectos.

Tema 4 Estructura de repetición

Usa bucles para situaciones reales, como iterar listas. Propón ejercicios con series matemáticas y procesos repetitivos. Fomenta la aplicación de bucles en la solución de problemas.

Tema 5 Métodos y parámetros

Usa ejemplos interactivos para enseñar métodos con parámetros. Propón retos donde los estudiantes escriban métodos con diferentes parámetros. Enfatiza la reutilización del código.

Tema 6 Arreglos

Utiliza gráficos para representar arreglos y sus índices. Crea ejercicios prácticos como invertir elementos o buscar máximos. Integra programación y lógica en las actividades.

Tema 7 Cadenas de caracteres

Enseña cómo manejar información textual con cadenas. Proporciona problemas prácticos como la búsqueda de patrones o la manipulación de textos. Muestra aplicaciones tangibles en programas reales.

Tema 8 Manejo de excepciones

Introduce el concepto de errores y cómo capturarlos. Propón escribir programas con errores intencionales para practicar. Usa escenarios interactivos que incluyan validación de datos.

Tema 9 Clases

Desglosa la estructura de una clase con ejemplos prácticos. Asigna proyectos donde los estudiantes creen clases completas. Usa juegos de roles para simular la interacción entre objetos.

Tema 10 Constructores

Explica constructores con ejemplos de inicialización automática. Proporciona ejercicios con constructores sobrecargados. Resalta la diferencia entre constructores vacíos y con parámetros.

Tema 11 Otros conceptos de claves y objetos

Amplía con ejemplos de clases abstractas e interfaces. Propón comparaciones entre herencia y composición. Fomenta debates sobre el uso adecuado de cada técnica.

Tema 12 Objetos que contienen objetos

Usa ejemplos visuales de objetos compuestos por otros objetos. Proporciona proyectos que usen objetos anidados. Incentiva el pensamiento estructurado y modular.

Tema 13 Colecciones

Introduce colecciones con ejemplos como listas de tareas o inventarios. Propón ejercicios para gestionar conjuntos dinámicos de datos. Enfoca en la búsqueda, inserción y eliminación dentro de colecciones.

Tema 14 Herencia

Explica la herencia con ejemplos de relaciones en la naturaleza. Propón ejercicios de jerarquías de clases para reducir duplicación de código. Refuerza la importancia de la reutilización del código.

Tema 15 Poliformismo

Usa ejemplos donde una acción toma diferentes formas según el objeto. Proporciona actividades que implementen polimorfismo en proyectos prácticos. Demuestra cómo reducir la complejidad y mejorar la eficiencia.



Evidencia 1

Descripción

Diagrama de clases básico para modelar la estructura de un problema y un programa escrito en Java mediante arreglos y strings.

Objetivo

Desarrollar un programa que emplea arreglos y strings para solucionar un problema.

Requerimientos

Software de programación

Instrucciones

1. Lee el siguiente caso:

Cajero Automático

Un banco desea introducir el servicio de Cajero Automático para proveer algunas facilidades a sus clientes.

Los cajeros solo pueden aceptar tarjetas de cuentas de cheques y tarjetas de cuentas de ahorros. Los usuarios solo pueden: ver su balance, retirar dinero de sus tarjetas y cambiar su PIN.

Cada tarjeta solo puede acceder a una cuenta, ya sea de cheques o de ahorros, pero, una cuenta de cheques puede ser accedida utilizando diferentes tarjetas (no más de 4 tarjetas). Un PIN está asociado con cada tarjeta para verificar la autenticidad del usuario.

Toda cuenta tiene un balance (cantidad de dinero que tiene la cuenta).

La cuenta de cheques tiene derecho a un 10% de sobregiro. Por lo tanto, para una cuenta de cheques, el máximo retiro permitido no debe exceder del balance más el sobregiro en el entendido de que en el cajero siempre habrá suficiente dinero.

Una cuenta de ahorros no tiene un porcentaje de sobregiro, por lo que el máximo retiro permitido no debe exceder del balance.

La información del cliente no se requiere en este sistema y se cuenta con 2 tipos de hardware (la lectora de tarjetas y el dispensador de dinero).

La lectora de tarjetas lee la tarjeta y envía el número al sistema. El sistema le indica a la lectora cuándo sacar la tarjeta.

2. Identifica las clases involucradas en el problema.
3. Especifica el tipo de relación que existe entre las clases.
4. Reconoce los atributos y comportamientos de cada clase. Incluir solamente los que se mencionan en el enunciado.

5. Determina el nivel de visibilidad de los atributos y métodos.
6. Construye el encabezado de cada método. Para este caso, no tomes en cuenta un lenguaje de programación, solo es importante determinar el nombre, qué parámetros debe recibir, si regresa o no un resultado.
7. Dibuja el diagrama de cada una de las clases identificadas.
8. Elaborar un programa que emplea arreglos y *strings* para solucionar un problema:

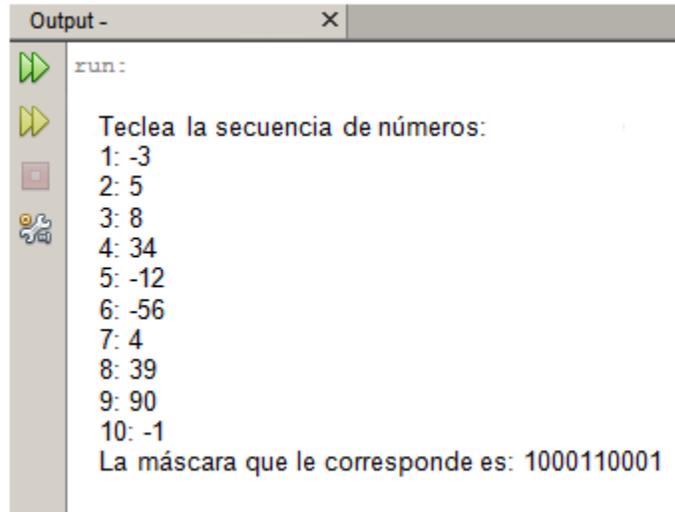
MANK

Dentro de un sistema computacional se requiere crear una "Máscara de ceros y uno" para una secuencia de números, de tal forma que se asigne un 0 para los valores positivos y el 1 para los negativos.

Por ejemplo: La secuencia: 1, -3, -5, 8, 9, -6, -43, -8 tendría la siguiente máscara: 01100111.

9. Escribe un programa que lea una secuencia de 10 números enteros y despliegue en pantalla la máscara.

Ejemplo de la ejecución del programa:



```
run:
Teclea la secuencia de números:
1: -3
2: 5
3: 8
4: 34
5: -12
6: -56
7: 4
8: 39
9: 90
10: -1
La máscara que le corresponde es: 1000110001
```

Restricciones del programa:

El programa debe contener un método de nombre *getMask* que regrese como resultado la cadena de caracteres que contiene la máscara.

Entregable

Documento electrónico que contenga el desarrollo de la actividad planteada.

Criterios de evaluación

Revisa la rúbrica correspondiente para conocer los criterios de evaluación.



Evidencia 2

Descripción

Aplicación computacional en lenguaje JAVA para solucionar un problema mediante el uso de clases y arreglos de objetos.

Objetivo

Desarrollar una aplicación que permita dar solución a la necesidad detectada con base en los requerimientos vistos en el desarrollo del curso.

Requerimientos

Software de programación

Instrucciones

1. Revisa la problemática del caso: **Videoclub *Games & Videos***

El dueño de un videoclub sabe de ti a través de una red de *networking*, te contacta y te informa que le interesa contratar tus servicios profesionales, pues te comenta que tiene un problema: necesita mejorar la administración de su negocio, contar con la elaboración de una aplicación para administrar las rentas de los dos tipos de productos que maneja; videojuegos y películas. Para todos los productos se tiene la siguiente información: nombre del producto, la cantidad a pagar por rentar ese producto, cantidad de días que se presta al momento de rentar y si en ese momento el producto está o no disponible. Para cada una de las películas se tiene el género (acción, fantasía, drama, comedio y aventura) y el año de producción. Para los juegos se cuenta con el estilo (acción, deportes y aventuras) y la plataforma (Xbox, Playstation, Wii) en la que se puede ejecutar.

2. Desarrolla una aplicación que permita realizar consultas y administrar las rentas de los productos como se describe más adelante:
 - a. Realiza el diagrama de clase de acuerdo con la descripción anterior. Debe incluir el nombre de la clase y las variables de instancia tanto de la clase base como de las clases derivadas.
 - b. Construye la clase base y las clases derivadas, las cuales deben incluir el constructor y los métodos **get** y **set** para cada una de las variables de instancia.
 - c. Construye una aplicación que emplee las clases que elaboraste en el punto anterior y que realice lo siguiente:
 - Leer la cantidad de películas que proporcionará el usuario y posteriormente leer la información de cada una de ellas:
nombre, costoRenta, días, genero, año, rentado
 - Leer la cantidad de videojuegos que proporcionará el usuario y posteriormente leer la información de cada uno de ellos:
nombre, costoRenta, días, estilo, plataforma, rentado
 - d. Permanecer en un ciclo ejecutando la acción solicitada por el usuario. La acción se solicita a través de algunos de los siguientes comandos:
 - T/t: Despliega todos los productos.
 - P/p: Despliega todas las películas.
 - V/v: Despliega todos los videojuegos.
 - S/s: Checar si una película está rentada.
 - J/j: Verifica si un videojuego está o no rentado.
 - C/c: Devuelve la cantidad de películas que están rentadas.
 - X/x: Indica la cantidad de videojuegos cuya plataforma es Xbox One.
 - U/u: Termina el programa.

Nota: el campo rentado se refiere a si el producto está o no rentado (1 está rentado, o no está rentado).

Entregables

Documento electrónico que contenga el desarrollo de la actividad planteada.

Criterios de evaluación

Revisa la rúbrica correspondiente para conocer los criterios de evaluación.



Competencia

Resuelve problemas con base en la elaboración y desarrollo de un programa que emplea arreglos y *strings* que le permitan dar solución a la necesidad identificada.

Nivel taxonómico: Utilización

Criterios de evaluación	Nivel de desempeño			
	Altamente competente 100%-86%	Competente 85%-70%	Aún sin desarrollar la competencia 69%-0%	
1. Identifica las clases y su relación.	20 - 18	17 - 14	13 - 0	20%
	1. Identifica todas las clases y establece una relación correcta entre ellas.	1. Identifica todas las clases. 2. Identifica una relación incorrecta entre las clases.	1. Identifica dos terceras partes de las clases y añade otras que no son significativas al problema. 2. No Identifica la relación entre las clases.	
2. Determina los atributos y los métodos de cada clase.	20 - 18	17 - 14	13 - 0	20%
	1. Identifica los atributos de todas las clases. 2. Identifica todos los métodos de todas las clases.	1. Identifica algunos de los atributos de todas las clases. 2. Identifica algunos de los métodos de todas las clases.	1. La mayoría de los atributos y métodos son incorrectos. 2. Identifica solo uno de los métodos de las clases.	
3. Construye	20 - 18	17 - 14	13 - 0	20%

el diagrama de clase.	<ol style="list-style-type: none"> 1. Incluye el diagrama para todas las clases. 2. Los atributos están descritos completamente. 3. Los métodos están descritos completamente. 4. La relación entre las clases está definida correctamente. 	<ol style="list-style-type: none"> 1. Incluye el diagrama para todas las clases. 2. Los atributos no están descritos completamente. 3. Los métodos no están descritos completamente. 4. La relación entre las clases está definida correctamente. 	<ol style="list-style-type: none"> 1. Se incluye el diagrama solo de algunas de las clases. 2. No define correctamente los atributos de la clase. 3. No define correctamente los métodos de la clase. 4. No marca correctamente la relación entre las clases 	
4. El programa soluciona la problemática.	<p>20 - 18</p> <ol style="list-style-type: none"> 1. El programa lee 10 números. 2. El programa indica qué número se está tecleando. El programa genera la máscara correcta. 	<p>17 - 14</p> <ol style="list-style-type: none"> 1. El programa lee 10 números. 2. El programa indica qué número se está tecleando. 3. El programa genera la máscara parcialmente correcta. 	<p>13 - 0</p> <ol style="list-style-type: none"> 1. El programa lee 10 números. 2. El programa no indica qué número se está tecleando. Despliega una máscara, pero no es la correcta. 	20%
5. El programa contiene un método que regresa la cadena resultante.	<p>20 - 18</p> <ol style="list-style-type: none"> 1. El programa tiene un método que obtiene el resultado. El método regresa un <i>string</i> con el resultado. 	<p>17 - 14</p> <ol style="list-style-type: none"> 1. El programa tiene un método que obtiene el resultado. 2. El método emplea <i>strings</i> parcialmente. El método despliega el resultado, no lo regresa. 	<p>13 - 0</p> <ol style="list-style-type: none"> 1. El programa contiene un método, pero se emplea para otras cosas. El método no emplea <i>strings</i> para generar el resultado. 	20%

Rúbrica de evidencia 2

Competencia

Resuelve problemas con base en la elaboración y desarrollo de un programa que emplea arreglos y *strings* que le permitan dar solución a la necesidad identificada.

Nivel taxonómico: Utilización

Criterios de evaluación	Nivel de desempeño			
	Altamente competente 100%-86%	Competente 85%-70%	Aún sin desarrollar la competencia 69%-0%	
1. Identifica las clases, la relación entre ellas, sus atributos y sus métodos.	20 - 18	17 - 14	13 - 0	20%
	1. Identifica todas las clases. 2. Establece una relación correcta entre las clases. 3. Detecta todos los atributos y métodos de cada una de las clases.	1. Identifica todas las clases. 2. No hay una correcta relación entre las clases. 3. Detecta mínimo 6 de los atributos y métodos de las clases.	1. No identifica todas las clases. 2. No hay una correcta relación entre las clases. 3. Detecta máximo 3 de los atributos y métodos de las clases.	
2. Construye el diagrama de clases.	20 - 18	17 - 14	13 - 0	20%
	1. El diagrama incluye todas las clases. 2. El diagrama establece la relación entre las clases. 3. El diagrama indica los atributos y métodos de cada clase.	1. El diagrama incluye todas las clases. 2. El diagrama establece una relación incorrecta entre las clases. 3. El diagrama indica los atributos y métodos de cada clase, pero no son colocados en la clase correcta.	1. El diagrama incluye al menos 2 de las clases. El diagrama no establece una relación entre las clases. 2. El diagrama no indica los atributos y métodos de cada clase.	
3. Elabora las	20 - 18	17 - 14	13 - 0	20%

<p>clases con los atributos y métodos correctos.</p>	<ol style="list-style-type: none"> 1. Elaboró todas las clases. 2. Cada clase incluye todos los atributos y métodos. 3. Se establece correctamente la relación entre las clases. 4. Los atributos y métodos tienen correctamente bien especificado el nivel de visibilidad. 	<ol style="list-style-type: none"> 1. Elaboró todas las clases. 2. Cada clase no incluye todos los atributos y métodos. 3. Se establece correctamente la relación entre las clases. 4. Los atributos y métodos no tienen definido correctamente el nivel de visibilidad. 	<ol style="list-style-type: none"> 1. Elaboró solo algunas de las clases. 2. Cada clase no incluye todos los atributos y métodos. 3. No se establece correctamente la relación entre las clases. 4. Los atributos y métodos no tienen definido correctamente el nivel de visibilidad. 	
<p>4. Elabora la aplicación la cual da de alta objetos de los diferentes tipos, lo cual se observa al desplegar la información de estos.</p>	<p style="text-align: center;">20 - 18</p> <ol style="list-style-type: none"> 1. La aplicación crea los arreglos de objetos. 2. La aplicación crea correctamente cada uno de los objetos del arreglo. 3. La aplicación despliega correctamente el contenido de cada uno de los arreglos. 	<p style="text-align: center;">17 - 14</p> <ol style="list-style-type: none"> 1. La aplicación crea los arreglos de objetos. 2. La aplicación crea correctamente solo un tipo de objeto. 3. La aplicación no despliega correctamente el contenido de cada uno de los arreglos. 	<p style="text-align: center;">13 - 0</p> <ol style="list-style-type: none"> 1. La aplicación crea algunos de los arreglos de objetos. 2. La aplicación no crea correctamente los objetos del arreglo. 3. La aplicación no despliega correctamente el contenido de cada uno de los arreglos. 	20%
<p>5. La aplicación permite realizar búsquedas en el arreglo de cada uno de los tipos de objetos dando respuesta a la petición del usuario especificada en un menú.</p>	<p style="text-align: center;">20 - 18</p> <ol style="list-style-type: none"> 1. La aplicación proporciona un menú de opciones. 2. La aplicación funciona correctamente para las 8 opciones proporcionadas en el menú. 	<p style="text-align: center;">17 - 14</p> <ol style="list-style-type: none"> 1. La aplicación proporciona un menú de opciones. 2. La aplicación funciona correctamente para las 5 opciones proporcionadas en el menú. 	<p style="text-align: center;">13 - 0</p> <ol style="list-style-type: none"> 1. La aplicación proporciona un menú de opciones. 2. La aplicación funciona correctamente para las 2 opciones proporcionadas en el menú. 	20%