Evaluación Parcial 2

Nombre: Herencia y colecciones

Sigla	Nombre Asignatura	Tiempo Asignado	% Ponderación
PGY2121	Desarrollo de Software y Escritorio	120 minutos	35%

1. Situación evaluativa

X	Ejecución práctica		Entrega de encargo		Prueba escrita		Presentación

2. Agente evaluativo

X	Heteroevaluación		Coevaluación		Autoevaluación

3. Tabla de Especificaciones

Resultado de Aprendizaje	Indicador de Logro (IL)	Ponderación Indicador Logro
RA3 Programa las colecciones que permitan almacenar y obtener información para dar solución a los	IL3.1 Aplica las sentencias de repetición, para ser representadas en la solución según el requerimiento del usuario en un caso de negocios.	5%
	IL3.2 Crea una colección del tipo ArrayList, para almacenar información según lo solicitado por el usuario en un caso de negocios.	5%

requerimientos de la organización de acuerdo con estándares de la industria.	IL3.3 Utiliza colecciones de tipos genérico, para dar solución a problemas del usuario en un caso de negocios.	15%
	IL3.4 Aplica los diferentes métodos que permiten agregar, modificar, eliminar y listar las colecciones, según lo solicitado por el usuario en un caso de negocios.	35%
RA4 Utiliza la herencia para implementar soluciones a los requerimientos de la organización de acuerdo con estándares de la industria.	IL4.1 Aplica los conceptos de herencia en la programación orientada a objetos que permitan dar solución a un problema planteado en un caso de negocios.	15%
	IL4.2 Programa los diferentes métodos solicitados en clases y subclases, permitiendo reutilizar código en un caso de negocios.	25%
	Total	**100%**

4. Instrucciones para el/la estudiante

Esta es una evaluación que corresponde a una ejecución práctica sin presentación y tiene un 35**%** de ponderación sobre la nota final de la asignatura. El **tiempo** para desarrollar esta evaluación es de **120 minutos** y se realiza de manera **individual** en **laboratorio pc avanzado.**

CASO

RentCar, se encuentra en una etapa de modernización de sus procesos, por lo cual ha decidido contratar sus servicios para que desarrolle un sistema que permita almacenar la información de sus vehículos en arriendo por categoría.

Los vehículos tienen un precio diario de arriendo (sin IVA) y cantidad de días que se arrendarán. Se clasifican en vehículos de carga y pasajeros. Los vehículos de carga tienen capacidad de carga en que se mide en kilos y los vehículos de pasajeros tienen la cantidad de pasajeros que pueden transportar. El desarrollo de la aplicación debe responder a los siguientes requerimientos:

- Cada vehículo debe poseer patente para identificarlo como único
- La clase padre debe ser abstracta.
- Las clases hijas no deben tener subclases.
- Debe existir un método para mostrar los datos del vehículo que debe ser sobrescrito dependiendo del tipo (carga o pasajero)
- Debe considerar generar por lo menos un constructor vacío y uno sobrecargado con todos sus datos en cada clase que se deba instanciar

- Debe implementar una interfaz con las siguientes constantes y un método
- IVA: 19%
- DESCUENTO_CARGA: 3%
- DESCUENTO_PASAJEROS: 7%

- Debe existir un método **en el archivo de interfaz de java** que permita mostrar el detalle de la boleta, considerando el valor de arriendo, el descuento, el IVA y el total

- Debe existir una clase auxiliar que permita almacenar los vehículos con las siguientes funcionalidades implementadas:
- Debe validar que el vehículo agregado no exista por su patente antes de guardar
- Un método que retorne la lista de vehículos
- Un método que permita obtener la cantidad de vehículos en que la cantidad de días de arriendo sea igual o superior a 7

Todas las clases deben tener los siguientes métodos implementados: constructores, accesadores y mutadores.

- En la clase ***main*** debe:
- Agregar un vehículo de carga y otro de pasajeros a la colección
- Listar los vehículos
- Mostrar las boletas de pago de cada vehículo
- Obtener la cantidad de vehículos cuya cantidad de días de arriendo sea igual o superior a 7

- **Recuerda** que se debe desarrollar el código aplicando todas las convenciones aprendidas en clases

FORMATO DE ENTREGA

CREAR LA CARPETA Nombre Apellido EN EL DISCO DE RESPALDO: DENTRO DE ELLA CONSTRUIR SU PROYECTO. NO OLVIDE COLOCAR SU NOMBRE EN EL COMENTARIO JAVADOC CORRESPONDIENTE

Pauta de Evaluación

Pauta tipo: Rúbrica

Categoría	% logro	Descripción niveles de logro
Muy buen desempeño	**100%**	Demuestra un desempeño destacado, evidenciando el logro de todos los aspectos evaluados en el indicador.
Buen desempeño	**80%**	Demuestra un alto desempeño del indicador, presentando pequeñas omisiones, dificultades y/o errores.
Desempeño aceptable	**60%**	Demuestra un desempeño competente, evidenciando el logro de los elementos básicos del indicador, pero con omisiones, dificultades o errores.
Desempeño incipiente	**30%**	Presenta importantes omisiones, dificultades o errores en el desempeño, que no permiten evidenciar los elementos básicos del logro del indicador, por lo que no puede ser considerado competente.
Desempeño no logrado	**0%**	Presenta ausencia o incorrecto desempeño.

Indicador de Evaluación		Categorías de Respuesta				Ponderación del Indicador de Evaluación
	Muy buen desempeño 100%	**Buen desempeño 80%**	**Desempeño aceptable 60%**	**Desempeño incipiente 30%**	**Desempeño no logrado 0%**	
Aplica las sentencias de repetición, para ser representadas en la solución según el requerimiento del usuario en un caso de negocios.	Utiliza las estructuras de repetición sin errores en la aplicación	Utiliza correctamente las estructuras de repetición existiendo un error lógico	Utiliza correctamente las estructuras de repetición existiendo dos errores lógicos	Utiliza correctamente las estructuras de repetición existiendo tres errores lógicos	Utiliza correctamente las estructuras de repetición existiendo más de tres errores lógicos o una de las sentencias impide la ejecución del programa	**5%**

Crea una colección del tipo ArrayList, para almacenar información según lo solicitado por el usuario en un caso de negocios.	Crea la colección en el registro para almacenar la información	----------	----------	----------	No crea la colección.	**5%**
Utiliza colecciones de tipos genérico, para dar solución a problemas del usuario en un caso de negocios.	Utiliza la colección para gestionar la información	----------	Utiliza la colección, pero faltan métodos por invocar	----------	No utiliza la colección.	**15%**
Aplica los diferentes métodos que permiten agregar, modificar, eliminar y listar las colecciones, según lo solicitado por el usuario en un caso de negocios.	Codifica correctamente en el registro, todos los métodos para gestionar la información que permitan dar solución al problema planteado.	Codifica correctamente el registro, pero falta uno de los métodos.	Codifica correctamente el registro, pero faltan dos métodos.	Codifica correctamente el registro, pero faltan tres métodos.	No codifica los métodos o faltan más de tres métodos.	**35%**
Aplica los conceptos de herencia en la programación orientada a objetos que permitan dar solución a un problema planteado en un caso de negocios.	Aplica todos los conceptos de herencia dentro de su modelo de clases, incluyendo la codificación de la interface para reutilizar para dar solución al problema planteado.	Aplica todos los conceptos de herencia dentro de su modelo de clases, pero no incorpora la interface le falta alguna constante y/o método abstracto.	Aplica todos los conceptos de herencia dentro de su modelo de clases, pero no incorpora la interface como recurso.	Incorpora sólo la interface como recurso, pero no la implementa dentro del sistema.	No aplica herencia ni incorpora interface.	**15%**
Programa los diferentes métodos solicitados en clases y subclases, permitiendo reutilizar código en un caso de negocios.	Programa todos los métodos en la clases y subclases, incluyendo los métodos abstractos de la interface y la	Programa todos los métodos en la clases y subclases, pero no incluye el uso de las	Programa métodos, pero falta uno de los ellos.	Realiza sólo los encabezados de los métodos, pero deja las	No programa los métodos.	**25%**

	utilización de las constantes.	constantes o le falta incorporar las instrucciones de los métodos abstractos en las clases hijas.		instrucciones en blanco.		
					Total	**100%**