U.N.M.d.P – F.I. - Ingeniería en Informática
Taller de Programación I
Validación y Verificación ADA

Grupo 4 - Integrantes:

- Noelia Echeverría
- Franco Ercoli
- Matías Gutierrez
- Juan Cruz Mateos

Parte 1 - Checklist

I – Desviación de los Objetivos				
#	I.1 Desviación	Si	No	N/A
1	¿El código implementa correctamente el diseño ?			✓
2	¿El código implementa más de lo que establece el diseño ?			✓
3	El mecanismo de envío (valor o referencia) de todos los parámetros de cada método es apropiado ?	✓		
4	Cada método retorna el valor correcto en cada punto de retorno ?	✓		
II – Omisión de Objetivos				
#	II.1 Omisión	Si	No	N/A
5	¿El código implementa completamente el diseño ?			✓
6	¿ Se han eliminado todos los restos de código innecesario o test de prueba en el código ?	✓		
III – Defectos en los Objetivos				
#	III.1 Declaración de Variables y Constantes	Si	No	N/A

7	¿ Los nombres de las variables y constantes son descriptivos y cumplen con las convenciones de nombres ?	✓		
8	¿ Los tipos de las variables son correctas ?	✓		
9	¿ Cada variables está inicializada apropiadamente ?	✓		
10	¿ Todas las variables que controlan ciclos (ciclos for) están declaradas en la cabecera del ciclo ?			✓
11	¿ Todas las variables y constantes están correctamente definidas ?	✓		
12	¿ El ámbito de las variables está correctamente definido ?	✓		
13	Todos los atributos tienen un indicador de acceso apropiado (private, protected, public)?		✓	
14	¿ Los atributos estáticos (static) son apropiados y viceversa ?	✓		
#	III.2 Definición de Métodos	Si	No	N/A
15	Los nombres de los métodos son descriptivos y cumplen con las convenciones de nombres ?	✓		
16	Todos los métodos tienen un indicador de acceso apropiado (private, protected, public) ?	✓		
17	El valor de los parámetros de cada método es chequeado antes de usarlo ?		✓	
18	¿ Los métodos estáticos (static) son apropiados y viceversa ?	✓		
#	III.3 Definición de Clases	Si	No	N/A
19	¿Cada clase tiene un constructor adecuado ?		✓	
20	¿ Las superclases contienen todos los miembros comunes de sus subclases ?	✓		
21	¿ La jerarquía de herencia de la clase está lo más simplificada posible?		✓	
#	III.4 Referencia a los Datos	Si	No	N/A

22	Para referenciar a un arreglo los valores de los subíndices están dentro del rango permitido ?			✓
23	Se verifica que toda referencia a un objeto o arreglo no sea nula ?		✓	
#	III.5 Expresiones y Tipos de Datos	Si	No	N/A
24	¿ Los cálculos se realizan con los tipos de datos consecuentes ?	✓		
25	Está contemplada la ocurrencia de overflow o underflow durante un cálculo ?		✓	
26	¿Por cada expresión se respeta el orden de evaluación y precedencia correcta ?	✓		
27	Se usan paréntesis para evitar ambigüedades ?	✓		
28	El código previene los errores por redondeo en forma sistemática			✓
29	¿El código evita sumas y restas sobre números con magnitudes muy diferentes ?			✓
30	Se chequea la división por cero o el ruido ?			✓
#	III.6 Comparación y Relaciones	Si	No	N/A
31	Las expresiones booleanas han sido simplificadas, usando “driving negations inward” ?	✓		
32	¿Cada prueba booleana chequea la condición correcta ?	✓		
33	Las comparaciones entre variables son de tipos inconsistentes ?	✓		
34	¿Son correctos los operadores de comparación ?	✓		
35	Todas las expresiones booleanas son correctas ?	✓		
36	¿No existen efectos colaterales inapropiados de una comparación ?	✓		

37	Los operadores lógicos no han sido confundidos con operadores a nivel bit?	✓		
38	¿El código evita la comparación de igualdad en números de punto flotante ?			✓
39	Están cubiertas las tres ramas de los if (menor,igual,mayor)		✓	
#	III.7 Control de Flujo	Si	No	N/A
40	¿Por cada ciclo se usa la mejor elección de construcción de ciclos ?	✓		
41	¿Todos los ciclos terminan ?	✓		
42	Cuando un ciclo tiene múltiples condiciones de salida todas están manejadas apropiadamente ?			✓
43	Todas las sentencias SWITCH tienen un caso por defecto ?			✓
44	Las salidas de un Switch no manejadas están debidamente comentadas y con una sentencia break ?			✓
45	¿Es correcta la profundidad en el anidamiento de ciclos ?			✓
46	Todo if anidado fue simplificado, en caso de ser posible, en sentencias SWITCH?		✓	
47	Los cuerpos nulos en las estructuras de control están marcados con llaves, marcados y comentados correctamente?	✓		
48	¿Todos los métodos terminan ?	✓		
49	Todas las excepciones son manipuladas apropiadamente ?	✓		
50	Las sentencias break con etiqueta derivan el control al lugar correcto ?			✓
#	III.8 Entrada/Salida	Si	No	N/A
51	¿Todos los archivos se abren antes de usarlos ?			✓
52	Los atributos de las sentencias de apertura de los archivos son consistentes con el uso de los mismos ?			✓
53	Todos los archivos se cierran cuando dejan de usarse ?			✓

54	Los datos en el buffer se envían al disco ?			✓

55	¿No hay errores de ortografía o gramática en el texto impreso o en la pantalla ?	✓		
56	Están chequeadas las condiciones de error ?	✓		
57	Se verifica la existencia de los archivos antes de intentar abrirlos ?			✓
58	Todas las excepciones de entrada/salida están razonablemente manejadas ?			✓
#	III.9 Interfaz del Módulo	Si	No	N/A
59	El número, orden, tipo y valores de parámetros en cada llamada de un método está de acuerdo con la declaración del método ?	✓		
60	Los valores respetan los acuerdos de unidades (por.ej., pulgadas versus yardas) ?	✓		
61	Si un objeto o arreglo es pasado a un método que lo altera, esta alteración es realizada correctamente por dicho método ?	✓		
#	III.10 Comentarios	Si	No	N/A
62	Todos los métodos, clases y archivos tienen los comentarios de cabecera apropiados ?		✓	
63	¿Cada atributo, variable o declaración de constante ha sido comentada ?		✓	
64	El comportamiento de cada método y clase es expresado en lenguaje plano ?		✓	
65	Los comentarios en la cabecera de cada método y clase son consistentes con el comportamiento del método o clase ?	✓		
66	Todos los comentarios son consistentes con el código ?	✓		
67	¿Los comentarios ayudan a entender el código ?	✓		
68	Hay suficientes comentarios en el código ?	✓		
#	III.11 Diseño y Empaquetado	Si	No	N/A

70	El formato estándar en el diseño e indentación del código es usado consistentemente ?	✓		
71	¿Algún método excede las 60 líneas ?		✓	
72	¿Algún módulo supera las 600 líneas ?		✓	
#	III.12 Modularidad	Si	No	N/A
73	Hay un bajo nivel de acoplamiento entre módulos (métodos y clases) ?	✓		
74	Hay un alto nivel de cohesión en cada módulo (métodos y clases) ?	✓		
75	No hay código repetido que se puede reemplazar por un método que implemente el comportamiento de dicho código ?	✓		
76	¿Se usan las librerías de clase java cuando y donde deben usarse ?	✓		
#	III.13 Almacenamiento	Si	No	N/A
77	Los arreglos tienen el tamaño suficiente ?	✓		
78	Las referencias a los objetos y arreglos son seteados a null una vez que dejan de usarse?		✓	
#	III.14 Perfomance	Si	No	N/A
79	Las estructuras de datos y algoritmos son lo más eficiente posible?		✓	
80	Los test lógicos están organizados, de manera que los más frecuentes y caros están primero ?	✓		
81	¿El costo de recálculo se encuentra lo más reducido posible mediante el almacenamiento de los resultados ?		✓	

82	Actualmente, se usa cada resultado calculado y almacenado ?	✓		
83	¿Los ciclos tienen los cálculos justos y necesarios ?	✓		
85	¿Ningún ciclo corto puede ser convertido en una estructura más simple ?	✓		

86	Dos ciclos distintos cumplen funciones necesariamente distintas que no pueden ser resueltas en simultáneo?	✓		
IV – Inconsistencia en los Objetivos				
#	IV.1 Performance	Si	No	N/A
87	¿Todos los códigos son implementados de modo consistente ?	✓		
V – Ambigüedad en los Objetivos				
#	V.1 Declaración de Variables y Constantes	Si	No	N/A
89	Todas las variables están definidas con nombres claros, consistentes y significativos ?	✓		
#	V.2 Performance	Si	No	N/A
90	No hay módulos excesivamente confusos que se pueden reestructurar o dividir en varias rutinas ?	✓		
VI – Redundancia en los Objetivos				
#	VI.1 Variables	Si	No	N/A
91	¿No existen variables o atributos redundantes o no usados ?	✓		
92	Toda variable está definida con en el alcance correcto	✓		
#	VI.2 Definición de Métodos	Si	No	N/A
93	¿No existen métodos que no son llamados o son innecesarios ?	✓		
#	VI.3 Perfomance	Si	No	N/A
94	¿Ningún código puede reemplazarse con llamadas a objetos externos reusables ?	✓		
95	No existen bloques de código repetidos que pueden condensarse en un método simple ?	✓		
96	¿Se eliminaron restos de código no usado o restos de rutinas de test ?	✓		
VII – Efectos Colaterales en los Objetivos				
#	VII.1 Definición de Métodos	Si	No	N/A

97	Después de cambiar un método se analizan los métodos que lo llaman	✓		
#	VII.2 Base de Datos	Si	No	N/A
98	El proceso de actualización y migración sigue el cambio de estructuras o contenidos en la base del proyecto ?			✓

Parte 2 - Recorridas tipo prueba de escritorio.

Clase TitularFactory:

Método:public static Titular getTitular(String tipo, String nombre, int dni, String tipoDePago)

N° Caso	Nombre variable	Tipo	Nombre	DNI	Tipo Pago	Resultado
1	titular1	Fisico	Pedro Gonzalez	22.563.333	Efectivo	✓
2	titular2	Juridico	EmpresaZ	11.588.963	Tarjeta	✓
3	titular3	Fisico	Carlos Perez	12589	Cheque	✓

Resultado: en todos los casos se obtienen los resultados esperados.

Clase DomicilioFactory:

Método: public static Domicilio getDomicilio(String direccion, String internet,
boolean celular, boolean telFijo,
boolean tv) throws DomicilioRepetidoException;

N° Caso	Dirección	Internet	Celular	telFijo	tv	Lanza Excepción	Resultado
1	Dorrego 1234	Internet100	true	false	false	no	✓
2	San Luis 1422	Internet500	false	true	false	no	✓

3	Paso 1598	Internet100	false	false	true	no	✓
4	Rivadavia 789	Internet500	true	true	true	no	✓
5	Cordoba 367	Internet100	false	false	false	no	✓
6	Cordoba 367	Internet500	false	false	false	si	✓

Resultado: en todos los casos se obtienen los resultados esperados.

Método: private static boolean repetido(String direccion);

N° Caso	Dir. Repetida	Dirección	return	Resultado
1	no	Formosa 134	false	✓
2	si	Formosa 134	true	✓

Resultado: en todos los casos se obtienen los resultados esperados.

Clase Titular:

Método: public void addDomicilio(String direccion, String internet, boolean celular, boolean telFijo, boolean tv);

N° Caso	Dirección	Internet	Celular	telFijo	tv	Imprime Repetido	Resultado
1	Dorrego 1234	Internet100	true	false	false	no	✓
2	San Luis 1422	Internet500	false	true	false	si	✓

Resultado: en todos los casos se obtienen los resultados esperados.

Obs: La prueba de escritorio del método addDomicilio se efectuó sobre la clase abstracta Titular debido a que es quien efectivamente tiene implementado el método (al que hacen referencia las clases concretas TitularFisico y TitularJurídico).

Clase TitularFisico:

Metodo: public Object clone();
Clonación profunda correcta.

Metodo: public double getCostoFinal();
El método calcula el costo final aplicando un descuento o un recargo según corresponda en función del tipo de pago, pero al no contar con la especificación de requerimientos no podemos validar que efectivamente la modificación sobre el costo final sea correcta.

Clase TitularJuridico:

Metodo: public Object clone() throws CloneNotSupportedException;
Siempre lanza la excepción.

Metodo: public double getCostoFinal();
El método calcula el costo final aplicando un descuento o un recargo según corresponda en función del tipo de pago, pero al no contar con la especificación de requerimientos no podemos validar que efectivamente la modificación sobre el costo final sea correcta.

Clase Empresa:

Método: public Titular solicitaDuplicado(int nroIdent);
El método funciona correctamente devolviendo el titular clonado o null según corresponda gracias a la correcta implementación del método clone() en las clases TitularFisico y TitularJuridico respectivamente.

Parte 3 - Propuestas sobre el código

Clase Empresa
Debido a que todos los métodos de la clase Empresa tienen como precondición "La empresa existe (no null)" sugerimos que sea un invariante de clase". Adicionalmente debemos mencionar que no tiene sentido la precondición "La empresa existe(no null)" sobre métodos de instancia (si la clase no fue previamente instanciada, estos métodos no pueden ser invocados).

Clase Contratacion
Sugerencia: el costo debería poder definirse o modificarse eventualmente (no hardcodeo). Agregar el método setCosto.

Clase DomicilioFactory
En el método getDomicilio, no está contemplado que el String internet no sea ni "Internet100" o "Internet500". Se sugiere o bien agregar como precondición en el javadoc del método, o bien, modificar la estructura interna del método con el objetivo de hacerlo extensible a otras velocidades de internet.

Clase TitularFactory

La precondición “La empresa existe (no es null)” no tiene sentido dentro del método debido a que nunca se utiliza una referencia a la empresa dentro del mismo.