NOMBRE:.....(Gestión/Sistemas)

1. (**0'5 ptos**) Establece la relación entre las clases `A`, `B` y `C` de manera que sea correcto el comportamiento asociado al siguiente código Eiffel (la letra griega que aparece entre llaves indica la implementación que se ejecuta).

```
oa:A; ob:B; oc:C
!!oa; !!ob; !!oc;
oa.f                    {α}
ob.f                    {β}
oc.f                    {δ}
oc.g                    {α}
oc.h                    {β}
oa:=oc
oa.f                    {α}
ob:=oc
ob.f                    {δ}
oa:=ob                  ILEGAL
```

2. En Java la clase `Stack` hereda de la clase `Vector` para poder hacer uso de las rutinas implementadas en ésta (***herencia de implementación***).

   a. (**0'5 ptos**) Si implementásemos estas dos clases en Eiffel ¿podríamos restringir el acceso a los métodos definidos en la clase `Vector` cuando el objeto receptor es un objeto `Stack`? ¿sería posible invocar a los métodos de la clase `Vector` sobre un objeto de la clase `Stack`?.

   b. (**0'5 ptos**) Explica cual sería la solución al problema propuesto en C++.

3. En Java la ***genericidad*** no es un elemento del lenguaje como ocurre en Eiffel.

   a. (**0'5 ptos**) Señala los inconvenientes que tiene el uso de una clase como `Stack` en Java frente a la clase `Stack[G]` en Eiffel.

   b. (**0'5 ptos**) Suponga la definición de una variable `prestamo` como una pila de personas, que nos va a servir para mantener el rastro de las personas por las que va pasando un libro determinado. Indica si sería correcto el siguiente código, sabiendo que el método `devolver` se encuentra definido en la clase `Persona`:

Eiffel	Java	
`prestamo: Stack [Persona];`	`Stack prestamo;`	`[1]`
`p1, p2:Persona;`	`Persona p1,p2;`	`[2]`
…		
`prestamo.push(p1);`	`prestamo.push(p1);`	`[3]`
`prestamo.push("Enrique");`	`prestamo.push("Enrique");`	`[4]`
`p2:=prestamo.pop;`	`p2=prestamo.pop();`	`[5]`
`p2.devolver();`	`p2.devolver();`	`[6]`

   c. (**0'5 ptos**) En la clase `Stack` de Java tenemos disponible el método `peek` que devuelve el objeto que se encuentra en el tope de la pila pero sin desapilarlo. Sabiendo esto, para evitar errores, ¿sería más adecuado sustituir las líneas [5] y [6] por la siguiente:

```
if (prestamo.peek() instanceof Persona) prestamo.pop().devolver();?
```

4. Sea el método `reintegro` definido en la clase `Cuenta` de acuerdo a la siguiente semántica:

```
reintegro(cantidad:Real) is
      require cantidad <= saldo
      do …
      ensure saldo = old saldo – cantidad
end
```

a. (**1 pto**) Explica la técnica del ***Diseño por Contrato*** y razona por qué las ***aserciones de Java*** no proporcionan el soporte adecuado a dicha técnica.

b. (**0'5 ptos**) En la implementación Java del método `reintegro` se ha decidido que se lance la excepción `IllegalArgumentException` en el caso de que la cantidad que se pase como parámetro sea mayor que el saldo de la cuenta. ¿De qué tipo será la excepción? Razona la respuesta.

c. (**0'5 ptos**) Sea el método correspondiente a una petición de la interfaz gráfica de usuario para cobrar un recibo, `cargoRecibo(Cuenta cta, double cantidad)`, que en un momento dado hace una llamada a la rutina `reintegro`. En el caso de que no se pueda realizar la operación se notificará lanzando una excepción `SaldoInsuficienteException`. ¿De qué tipo será esta excepción? Razona la respuesta.

d. (**0'5 ptos**) Escribe el "esqueleto" de las rutinas anteriores para mostrar el manejo y definición de las excepciones `IllegalArgumentException` y `SaldoInsuficienteException`.

5. Sea una clase diferida `Almacenable` que contiene la definición de un único método diferido `escribe` y una variable de instancia de tipo `String almacen`, con el nombre del almacén que se va a utilizar. Esta clase es la clase padre de las clases `Transmisor` y `Receptor` que implementan el método `escribe`. En el caso del transmisor envía información desde el almacén de datos y en el caso del receptor almacena información en el almacén. Por último, se ha implementado una clase `Radio` que hereda tanto de la clase `Transmisor` como de la clase `Receptor` en la que la implementación del método `escribe` llama a la implementación del método con el mismo nombre en las clases padre y además hace otras acciones específicas de la clase `Radio`.

   a. (**0'75 ptos)** ¿Son las clases diferidas un elemento del lenguaje C++? Razona la respuesta y proporciona el código C++ para la clase `Almacenable`.

   b. (**0'75 ptos**) Especifica en C++ la jerarquía de herencia que se detalla en el ejercicio. (NOTA: no hay que implementar el método escribe de `Transmisor` y `Receptor`).

6. Siguiendo la definición de la clase `LINEAL_ITERATOR[G]` de Eiffel vista en clase:

   a. (**0'5 ptos**) ¿Es correcta la implementación del método `forEach` (ejecuta una misma acción sobre todos los elementos de una colección) que se da a continuación? Justifica la respuesta:

```
forEach is do
      from coleccion.start
      until coleccion. after
      loop
            coleccion.item.action
            coleccion. forth
      end
end;
```

   b. (**0'5 ptos**) A partir de la definición de la clase `LINEAL_ITERATOR[G]` implementa un iterador `ITERADOR_ASCENSO` que se encargue de subir el sueldo a todos los empleados de una empresa. El conjunto de empleados se guarda en una colección lineal que se le pasará al iterador en el momento de la creación.

   c. (**0'5 ptos**) Atendiendo a la clasificación de los iteradores que los divide en iteradores internos y externos explica a qué grupo pertenecería el iterador `ITERADOR_ASCENSO`.

   d. (**0'5 ptos**) Explica la importancia de las clases parcialmente diferidas para conseguir código genérico. Utiliza la implementación del método `forEach` de Eiffel para apoyar tu explicación.

   e. (**1 pto**) Supongamos que la `LinkedList` de Java incluye el método `forEach` que recorre la colección y ejecuta sobre cada uno de los elementos la acción que se le indica en el momento de invocarlo. Implementar el método `forEach` y todo lo que sea necesario para utilizarlo para subir el sueldo a todos los empleados.

      NOTA: `LinkedList` tienen disponibles los métodos `size()` que devuelve el número de elementos de la colección y el método `get(int index)` que devuelve el objeto de la posición `index`. También el método `iterator()` que devuelve un objeto `Iterator`.