Universidad de Jaén
Departamento de Informática

Programación Orientada a Objetos

Seminario de preparación Primera prueba práctica

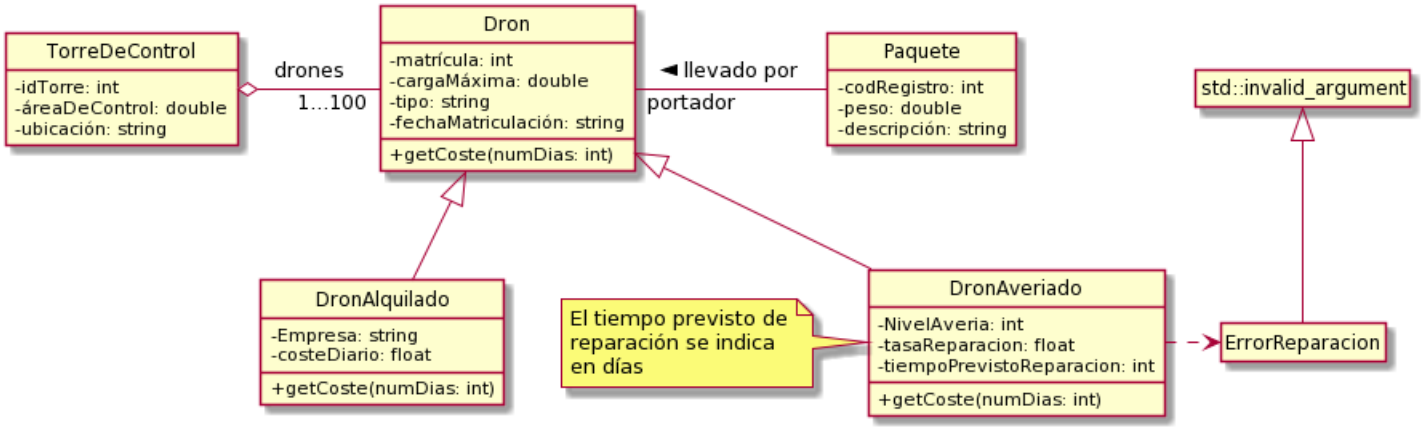
http://bit.ly/ujapoo

Instrucciones

- **Dispone de 1 hora y 40 minutos** para la realización de los ejercicios propuestos.
- **Material:** Sólo puede hacer uso para la realización de la actividad del material de teoría y guiones de prácticas accesible en el espacio virtual de la asignatura. También dispone de código fuente de algunas partes del problema que debe completar siguiendo los ejercicios propuestos. El profesor le aclarará la sintaxis de cualquier duda que tenga sobre cualquier función o clase de la biblioteca estándar de C++ que necesite para la implementación siempre y cuando considere que es necesaria para su realización. Para la edición del código, compilación y depuración debe utilizarse el ordenador del aula con cualquiera de los entorno de desarrollo y compiladores instalados.
- **Recomendaciones**: grabe continuamente el código generado en disco para evitar pérdidas de información en caso de bloqueo de la máquina. Si esto ocurriera, notifíquelo al profesor para que le indique cómo proceder. Vaya implementando, compilando y probando cada ejercicio de forma individual. Si no sabe cómo implementar algún método, implemente al menos su cabecera y deje el contenido en blanco o comentado. Al menos podrá llamarlo desde donde proceda y continuar realizando el resto de ejercicios. Si encuentra errores de compilación que no es capaz de resolver, comente el fragmento de código que genere los problemas y siga trabajando con el resto de ejercicios.
- **Está prohibido expresamente** utilizar cualquier otro material impreso, en formato digital o accesible de forma on-line para la realización del ejercicio. Tampoco se pueden realizar copias del presente enunciado o del material realizado en cualquier formato ni sacarlas del aula utilizando cualquier soporte de almacenamiento o servicio telemático. Cualquier incumplimiento de estas normas supondrá el suspenso inmediato del ejercicio.
- Al finalizar la actividad, **entregue todos los ficheros de código fuente y el último ejecutable obtenido**, comprimidos en un fichero en **formato .ZIP** a través del espacio virtual de su grupo de prácticas en la actividad creada a tal efecto. **Entregue también esta hoja** debidamente cumplimentada.
- *No olvide borrar del ordenador todo el código realizado al finalizar su trabajo* y después de asegurarse de que lo ha enviado correctamente. Si tiene dudas al respecto, pregúntele al profesor si ha realizado correctamente la entrega.
- *Podrá acceder a todo el material que entregue así como a este enunciado a partir del día siguiente a la realización de esta actividad* en el espacio virtual de su grupo de prácticas.

Descripción de la actividad

Intelig∀ está satisfecha con tu trabajo, y ahora te pide que incluyas también información sobre qué drones son de alquiler, así como si alguno reporta alguna avería en vuelo, y su estado. El nuevo diseño sobre el que tienes que trabajar queda así:



Ejercicios

NOTA: si el código no compila, se penalizará un 10% de la nota que se obtenga en la prueba.

1. *(2 puntos)* Define e implementa las clases **DronAveriado** y **DronAlquilado** siguiendo el diagrama de clases anterior. Incluye los métodos *get* y *set* para los atributos de tipos simples, así como un constructor parametrizado en cada una de ellas.

   > **Prueba 1** *(0.5 puntos)* En la función *main*, crea dos objetos de la clase **DronAveriado** y dos objetos de la clase **DronAlquilado** y añadelos a la torre control *t1*. Captura y procesa las excepciones que se puedan lanzar.

2. *(1 punto)* Implementa los constructores de copia de las clases **DronAveriado** y **DronAlquilado**.

3. *(1.5 puntos)* Cada dron tiene en condiciones normales un coste fijo de 20€ al día. Desde la torre de control se necesita conocer el coste que supone la reparación de un dron averiado y el coste de cada dron alquilado. Para calcular el coste de la reparación se tiene que aplicar la siguiente fórmula: *Coste = tasa * nivel de avería * número de días*. Para calcular el coste del alquiler se realiza el siguiente cálculo: *Coste = coste diario * número de días*. **Redefine** el método *GetCoste* de la clase **Dron** en sus clases derivadas de manera que se cumplan estas condiciones, y **adapta** la clase **Dron** para que se ejecute el método correcto en cada situación.

4. *(1.5 puntos)* Define e implementa la clase **ErrorReparación** tal y como se indica en el diagrama de clases. Lanza una excepción del tipo **ErrorReparación** en el constructor parametrizado de la clase **DronAveriado** si el tiempo de reparación es inferior a cero.

   > **Prueba 2** *(1 punto)* En el archivo *main.cpp*, implementa la función *VisualizaDrones (TorreDeControl& param)* que muestre el tipo (si es normal, alquilado, o si está averiado), la matrícula y el coste diarios de los drones de una torre de control. En la función *main*, usa esta función para mostrar los datos de los drones de la torre de control *t1*.

5. (2 puntos) Implementa el método *TorreDeControl::buscaDron(float coste)*. Este método ha de devolver **la dirección de memoria** del dron de la torre de control cuyo coste diario sea menor, siempre que este coste esté por debajo del valor que se pasa como parámetro. Si ningún dron cumple esta condición, se devolverá *nullptr*.

   > **Prueba 3** *(0.5 puntos)* En la función *main*, utiliza el método *TorreDeControl::BuscaDron* para encontrar al dron más económico de la torre *t1*, con un coste diario inferior a 50€, y muestra la matrícula y su coste por consola.