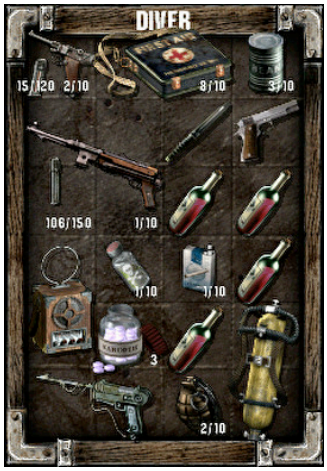
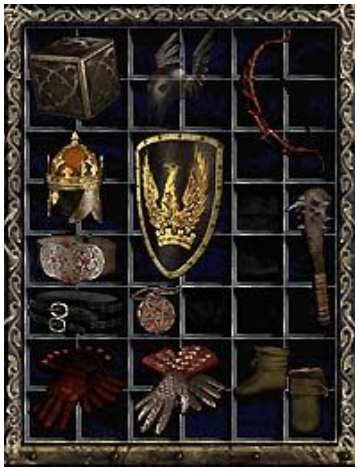
Práctica 3. Módulo Algoritmo Voraz.

En la mayoría de videojuegos se permite al jugador guardar los objetos que se encuentra a lo largo de la aventura dentro de un inventario. Algunos juegos ponen un peso límite o un número máximo de objetos que se pueden llevar. Pero hay otros, como Commandos o Diablo, en los que el inventario es una cuadrícula, tal y como se muestra en las imágenes, en las que la cantidad de objetos que se pueden llevar viene limitada por la forma, tamaño y posición de los mismos.





Para esta práctica, vamos a tomar como modelo de inventario una cuadrícula de tamaño fijo de 5 x 5, en la que podemos tener objetos con tamaños (**ancho x alto**) de 1x1, 2x1, 1x2, 2x2, 3x1, 1x3 y 3x2. Además del tamaño, los objetos tienen un **identificador** y un **valor** positivo en función de su utilidad. Por ejemplo, en el caso de objetos de tamaño 1x1 (imagen izquierda) valdrá más la pistola que la botella de vino.

La forma de introducir un objeto en el inventario debe hacerse buscando un hueco suficientemente grande para albergarlo, recorriendo el array de izquierda a derecha y de arriba abajo y sin rotarlo. Por ejemplo, consideremos el inventario tras entrar los objetos con id 0 (3x2), 5 (1x2) y 7 (2x2), mostrado en la imagen izquierda. Si el siguiente en entrar es un objeto con id 1 de 1x3 (imagen central) y luego el objeto con id 2 de 3x2, quedaría como se muestra en la imagen derecha.

0	0	0	5	
0	0	0	5	
7	7			
7	7			

⇒

0	0	0	5	**1**
0	0	0	5	**1**
7	7			**1**
7	7			

⇒

0	0	0	5	**1**
0	0	0	5	**1**
7	7			**1**
7	7	**2**	**2**	**2**
		2	**2**	**2**

Podría suceder que queden huecos en la solución.

SE PIDE:

Desarrollar un algoritmo en Java basado en el esquema voraz que encuentre una solución que maximice el llenado del inventario en función del valor de utilidad de los objetos.

- Implementar la función *seleccionarCandidato()* se evaluará con hasta 4 puntos.
- Implementar la función *esCandidatoFactible()* se evaluará con hasta 4 puntos.
- Implementar el algoritmo voraz *llenarInventario()* se evaluará con 1 punto.
- Entregar una memoria explicando el funcionamiento de las 3 funciones anteriores y alguna otra auxiliar, si se ha necesitado, se evaluará con 1 punto.

No es necesario que el algoritmo voraz asegure la solución óptima, pero sí una buena aproximación.

Nota: Es requisito para la corrección de la práctica que el programa entregado compile sin errores, funcione correctamente y siga el esquema voraz.

DESARROLLO:

Se proporcionan cuatro clases: *Objeto* (clase que contiene las características de un objeto), *Inventario* (clase donde se guardan los objetos), *Pruebas* (clase lanzadera para probar el algoritmo) y *Principal* (clase donde hay que implementar el algoritmo solicitado).

Objeto.Java

Esta clase contiene los atributos y métodos necesarios para manejar un objeto. *No se puede modificar.*

Inventario.Java

Esta clase contiene los atributos y métodos necesarios para manejar el inventario. *No se puede modificar.*

Pruebas.Java

Esta clase proporciona una lanzadera para realizar las pruebas con las que poder verificar que la codificación realizada por el alumno cumple con los requisitos. Se puede modificar para incluir cuantas pruebas desee hacer el alumno. *No se entrega*.

Principal.Java

Esta clase contiene las cabeceras de los métodos a implementar en la práctica.

public static ArrayList<Objeto> llenarInventario(ArrayList<Objeto> candidatos,
Inventario inventario)

La cabecera del algoritmo voraz. Recibe una lista dinámica con los candidatos y un inventario. Devuelve una lista dinámica con los objetos que entran en el inventario y el propio inventario relleno con esos objetos.

private static Objeto seleccionarCandidato(ArrayList<Objeto> candidatos)

La cabecera del método que permite seleccionar un candidato. Recibe una lista dinámica con los candidatos y devuelve el objeto seleccionado.

private static boolean esCandidatoFactible(Objeto candidato,
Inventario inventario)

La cabecera del método que comprueba si un candidato es factible o no y, en caso afirmativo, lo incluye en el inventario. Recibe un objeto y un inventario. Devuelve **true** si el objeto es válido para entrar en el inventario; **false**, si no lo es.

NORMAS DE ENTREGA:

La entrega consistirá en subir a la plataforma Moodle un archivo en formato comprimido (ZIP o RAR) que incluya un fichero PDF con la memoria y el fichero Principal.java donde se encuentre implementado las funciones del algoritmo.

Si la práctica se ha desarrollado individual, el nombre del fichero comprimido deberá ser el número de matrícula del alumno (ejemplo: ai0260.zip). Si la práctica se ha desarrollado en pareja con otro compañero, el nombre del fichero comprimido deberá ser el número de matrícula de ambos (ejemplo: ai0340-bc0433.zip). En este último caso, **AMBOS** alumnos deberán subir la práctica al Moodle.

El archivo comprimido se deberá subir a la plataforma Moodle **antes de las 23:55 horas del 17 de noviembre de 2019**. Aquellos alumnos que hayan subido la práctica deberán realizar un examen escrito de la misma el día **25 de noviembre de 2019**. Para poder aprobar esta práctica será indispensable haberla subido a la plataforma, haber superado el examen de la misma y haberla desarrollado correctamente.