Práctica

1

PRINCIPIOS DE LA INGENIERÍA DEL SOFTWARE

CURSO 2009-2010

Universidad Carlos III de Madrid

Primera Práctica 2009/2010

Universidad Carlos III De Madrid
Principios de Ingeniería Informática
María Isabel Sánchez Segura
José Arturo Mora-Soto
Juan Carlos Alonso Durán

LOS PROCESOS PERSONALES

Práctica 1

Universidad Carlos III de Madrid. Escuela Politécnica Superior

Universidad Carlos III De Madrid
Principios de Ingeniería Informática
María Isabel Sánchez Segura
José Arturo Mora-Soto
Juan Carlos Alonso Durán

Capítulo

1

OBJETIVO DE LA PRÁCTICA

Aprender a identificar y definir procesos de desarrollo de software.

PLANTEAMIENTO DE LA PRÁCTICA

En el curso de Programación I del cuatrimestre anterior se desarrolló un programa en Java inspirado en el juego tradicional de la Batalla naval o Juego de los barquitos. En este juego de dos jugadores, cada uno de ellos dispone de una flota formada por barcos de diferentes tamaños que debe distribuir, de manera secreta, sobre una cuadrícula bidimensional que representa un área del mar. Una vez que ambos jugadores han terminado de colocar todos sus barcos, uno de ellos, elegido aleatoriamente (Jugador 1), "disparará" sobre el área de mar controlada por el otro jugador (Jugador 2), indicando para ello las coordenadas del disparo, con la intención de impactar a uno de sus barcos. El jugador atacado (Jugador 2) declarará si el disparo ha caído en una casilla vacía (Agua) o ha impactado en uno de sus barcos (Tocado). A continuación será el Jugador 2 el que realice el disparo sobre la zona controlada por el Jugador 1. Los jugadores continuarán realizando disparos de forma alternativa hasta que resulten hundidos todos los barcos de uno de los dos jugadores, momento en el que termina el juego.

Tomando como base el programa de la Batalla naval desarrollado, o en su defecto algún otro programa que haya elaborado, describa los procesos que llevó a cabo para su desarrollo, desde la lectura y comprensión del enunciado hasta la entrega final del programa ejecutable. La descripción de los procesos se debe realizar aplicando la notación de descripción de procesos ETVX (Entry Task Verification eXit).

MODELADO DE PROCESOS CON ETVX

El modelo ETVX es una notación (Figura 1) que permite describir los procesos y sus componentes los cuales son:

- (E = Entry) Criterios de entrada: Son aquellas condiciones que deben cumplirse antes de empezar el proceso.
- (T = Tasks) Tareas: Conjunto de descripciones que indican qué debe realizarse en el actual proceso.
- (V = Verification) Criterios o tareas de validación: Procedimientos para verificar la calidad de los elementos de trabajo producidos.
- (X = eXit) Criterios de salida: Son aquellas condiciones que deben cumplirse al término del proceso.

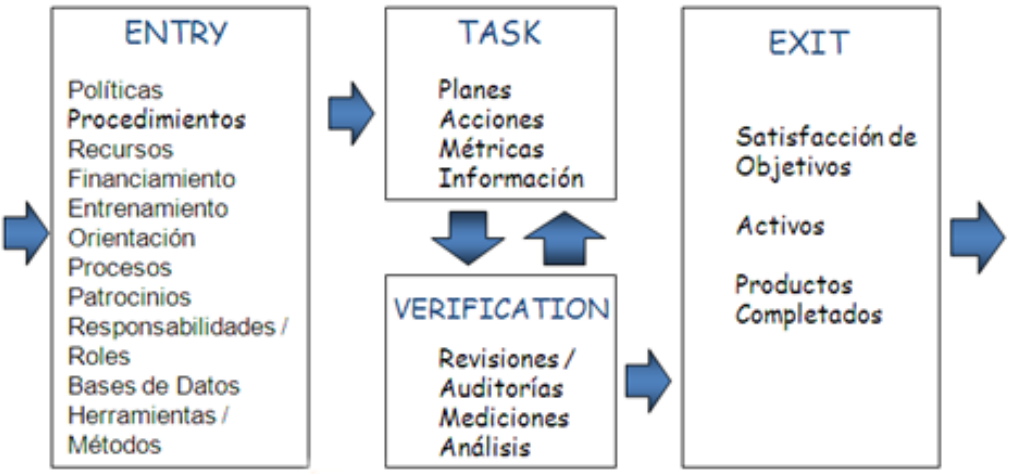


Figura 1. Notación ETVX

Del mismo modo, es posible alternativamente usar una representación tabular de ETVX como se muestra a continuación:

Criterios de Entrada	Tareas	Verificación	Criterios de Salida

A su vez cada una de las tareas puede ser modelada utilizando estos elementos, hasta alcanzar un nivel de detalle que sea simple y fácil de entender. De esta forma, es posible obtener varios niveles de especificación del proceso.

Universidad Carlos III De Madrid
Principios de Ingeniería Informática
María Isabel Sánchez Segura
José Arturo Mora-Soto
Juan Carlos Alonso Durán

Ejemplo de un proceso de envío de un mensaje de texto por medio de un móvil descrito con ETVX:

Criterios de Entrada	Tareas	Verificación	Criterios de Salida
• Móvil encendido • Móvil con cobertura • Existe saldo suficiente para realizar el envío	• Habilitar panel para crear nuevos mensajes • Seleccionar el tipo de mensaje a enviar (texto, audio) • Seleccionar o Ingresar Nombre del contacto destino • Ingresar texto del mensaje • Pulsar opción "Enviar"	• El nombre o número del contacto destino ha sido ingresado correctamente y existe • El mensaje de "envío" ha sido mostrado	• Mensaje de texto enviado • Saldo actualizado

Ejemplo del proceso PostMortem de PSP0 descrito con ETVX:

Criterios de Entrada	Tareas	Verificación	Criterios de Salida
• Descripción del problema y declaración de requisitos • Formulario del resumen del Plan del proyecto con los datos del tiempo de desarrollo • Logs de registro de Defectos y Tiempos completados • Un programa probado y ejecutado.	• Registro de Defectos Pulsar opción "Enviar" • Consistencia de los datos defectuosos • Revisar y corregir los datos de tiempo	• Los datos son recolectados en un formato estructurado y definido • Los datos son completos y exactos	• Programa probado completamente • Formulario del Resumen del Plan del Proyecto completado • Logs de Registro de Defectos y de Tiempos completado

Universidad Carlos III De Madrid
Principios de Ingeniería Informática
María Isabel Sánchez Segura
José Arturo Mora-Soto
Juan Carlos Alonso Durán

Capítulo

2

Normas y Procedimiento para la entrega de la Práctica

La práctica 1: **Los Procesos Personales**, se realizará en **grupos de 2 personas**, debiéndose comunicar los integrantes de cada grupo al instructor con la mayor antelación posible. El documento a entregar se realizará mediante Aula Global. El asunto del mensaje debe ser el nombre de la práctica y como texto del mensaje deberán figurar nombre, apellidos y NIA de los alumnos que realizan la entrega.

Entregables

Generar un documento Word con el modelo de procesos ETVX para ser enviado al instructor.

Se establece una única entrega cuya fecha está definida de acuerdo al programa curricular del curso.

De acuerdo a las normas de la asignatura el hecho de no entregar esta práctica dentro del plazo de entrega supondrá una calificación de 0 en la misma.

Criterios de Evaluación

El documento debe contener para cada proceso identificado los elementos de la notación ETVX: Criterios de entrada, tareas, verificación y criterios de salida.

Los procesos deben estar especificados a un nivel de detalle que permita el claro entendimiento del mismo.

Escribir el documento con una redacción clara.

El documento debe ser entregado a más tardar en la fecha solicitada.

Sugerencias

Cada alumno debe conservar, hasta el final de la asignatura, una copia de la información enviada.

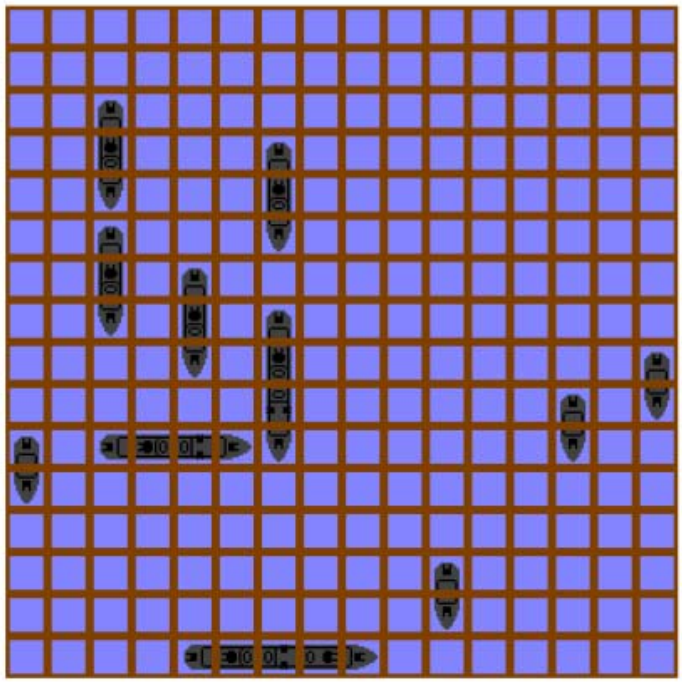
Anexo

Enunciado juego de la Batalla Naval

1. Introducción

El objetivo de la tercera práctica del curso es que los alumnos desarrollen un programa Java inspirado en el juego tradicional de la Batalla naval o Juego de los barquitos. En este juego de dos jugadores, cada uno de ellos dispone de una flota formada por barcos de diferentes tamaños que debe distribuir, de manera secreta, sobre una cuadrícula bidimensional que representa un área del mar. Una vez que ambos jugadores han terminado de colocar todos sus barcos, uno de ellos, elegido aleatoriamente (Jugador 1), "disparará" sobre el área de mar controlada por el otro jugador (Jugador 2), indicando para ello las coordenadas del disparo, con la intención de impactar a uno de sus barcos.
El jugador atacado (Jugador 2) declarará si el disparo ha caído en una casilla vacía (Agua) o ha impactado en uno de sus barcos (Tocado). A continuación será el Jugador 2 el que realice el disparo sobre la zona controlada por el Jugador 1. Los jugadores continuarán realizando disparos de forma alternativa hasta que resulten hundidos todos los barcos de uno de los dos jugadores, momento en el que termina el juego.

En Internet se pueden encontrar numerosas versiones de este juego. El alumno puede probar alguna de estas versiones para entender mejor el desarrollo del juego y lo que se pretende realizar en las prácticas de este curso.



2. Objetivos docentes

Los objetivos docentes de la primera práctica son:

- Familiarizar al alumno con el entorno gráfico de programación Eclipse. El alumno deberá ser capaz de instalar el kit de desarrollo Java J2SE1.6 y el entorno Eclipse y hacerlos funcionar adecuadamente.
- Hacer que el alumno conozca la sintaxis de Java: tipos de datos, instrucciones, ficheros que genera, etc.
- Capacitar al alumno para la creación de un programa en Java, con sus correspondientes métodos y atributos, así como para la solución de problemas utilizando la metodología de programación estructurada.

3. Trabajo a realizar

3.1 Representación del tablero de juego

- El tablero está formado por zonas (o subtableros). Cada uno de los dos jugadores debe contar con dos zonas, una para representar la situación de la flota propia y otra para representar la situación de la flota enemiga.
  - Cada zona tendrá unas dimensiones de 20 filas x 26 columnas.
  - Las filas estarán numeradas del 1 al 20.
  - Las columnas serán designadas con letras, de la A a la Z.
- Cada jugador dispone de los siguientes barcos:
  - 1 Portaviones (ocupa 5 casillas).
  - 2 Acorazados (ocupa 4 casillas cada uno).
  - 3 Destructores (ocupan 3 casillas cada uno).
  - 4 Submarinos (ocupan 2 casillas cada uno).

Representación en pantalla

- Se utilizará una cadena de caracteres para mostrar por pantalla la situación de las zonas del tablero pertenecientes a un jugador (Zona Propia y Zona Enemiga):
  - “#” identificará a las zonas de los barcos que no han resultado dañadas.
  - “T” identifica a las zonas de los barcos que han resultado dañadas por un disparo. Cuando el barco esté hundido, todo él pasará a estar representado por “@”
  - “ ” (un espacio en blanco) identificará a las casillas de AGUA (vacías), que no contienen ningún barco y no han recibido ningún disparo.
  - “.” identificará a las casillas de agua que han sido alcanzadas por un disparo.
- A continuación se presenta un ejemplo de salida por pantalla:

```
TURNO 22 DEL JUGADOR 1

      ---* JUGADOR 1 *---
           ZONA PROPIA
   ABCDEFGHIJKLMNOPQRSTUVWXYZ
01     .   .                 01
02    #          ##         . .02
03    #     #           .    .  03
04    #     #          #    .    04
05    #   . #          #         05
06                     #    ##   06
07                     #         07
08    .      ##        #        #08
09                .    .        #09
10              .               . @10
11            .               #   11
12                            #   .12
13                  .         #   13
14                   #        #   14
15     .             #            15
16         . ##      #            16
17                                17
18    .         .                 18
19                                19
20              .   .             20
   ABCDEFGHIJKLMNOPQRSTUVWXYZ


           ZONA ENEMIGA
   ABCDEFGHIJKLMNOPQRSTUVWXYZ
01     .   .                    01
02                             .02
03                      .   ..  03
04                         .    04
05       .                     05
06                             06
07                             07
08    .                        08
09                 .           09
10              .              .10
11            .                11
12                             .12
13                   .         13
14                             14
15     .                       15
16         .                   16
17                             17
18    .         .              18
19                             19
20              .   .          20
   ABCDEFGHIJKLMNOPQRSTUVWXYZ

                    Puntuación: -21
```

Trabajo a realizar

- Crear una clase denominada BatallaNaval, en la que se creará un tablero de juego y se permitirá jugar al juego de la batalla naval.

Universidad Carlos III De Madrid
Principios de Ingeniería Informática
María Isabel Sánchez Segura
José Arturo Mora-Soto
Juan Carlos Alonso Durán

- Crear un método main y dentro de él crear uno o varios arrays de char para representar el tablero de juego (2 tableros para cada jugador).
- Crear constantes que representen el número de unidades de cada tipo de barco del que se dispone al principio de la partida.
- Crear una variable para representar el turno en el que nos encontramos y otra para representar la puntuación.
- Inicializar los 4 tableros de forma que todas sus casillas tengan como valor AGUA.
- Colocar los barcos en nuestro tablero y en el tablero enemigo.
- Imprimir por pantalla los 4 tableros según el ejemplo anterior.

3.2 Implementación del Juego

Método *inicializarTablero*

- En primer lugar, se inicializará el tablero de juego (todas las casillas de las dos zonas del tablero de cada uno de los dos jugadores serán inicializadas como AGUA).
- Para ello, se creará el método inicializarTablero que reciba como parámetro las zonas a inicializar y no devuelva nada. Se recomienda reutilizar el código creado en el apartado anterior.

Método *mostrarTablero*

- Se mostrarán por pantalla las zonas del tablero del jugador humano como una cadena de caracteres, es decir:
  - Zona Propia: en la que aparecen los barcos propios con sus distintas partes dañadas (‘@’) o no (‘#’), y zonas de agua que han recibido disparos (‘.’) o no (‘ ’).
  - Zona Enemiga: en la que únicamente aparecen las casillas sobre las que el jugador ha efectuado disparos, que hayan alcanzado un barco (‘T’ o ‘@’) o una casilla de agua (‘.’); las casillas sobre las que no se han efectuado disparos todavía serán representadas con (‘ ’).
- Para ello, se creará el método mostrarTablero que reciba como parámetro el tablero de juego, el código de jugador, el turno y su puntuación. No devolverá nada.

Método *seleccionarNumeroPersonas*

- Antes de empezar el juego, se preguntará al usuario por teclado si cada uno de los dos jugadores será controlado por un jugador humano o el ordenador (de manera que habrá tres posibles tipos de partida: jugador humano vs. jugador humano; jugador humano vs. ordenador; y ordenador vs. ordenador).
- Para ello, se creará el método seleccionarNumeroPersonas que no reciba parámetros de entrada y devuelva un dato entero indicando la opción seleccionada por el usuario, según el siguiente criterio:
  - 0: Ambos jugadores controlados por el ordenador.
  - 1: Jugador humano contra ordenador.
  - 2: Ambos jugadores humanos.

Método *distribuirBarcos*

- Posteriormente los jugadores colocarán sus barcos.
  - A los jugadores humanos se les ofrecerá la alternativa de colocar sus barcos automática o manualmente, preguntándoselo por teclado.
  - La colocación de los barcos de los jugadores controlados por el ordenador será siempre automática.
  - Para ello, se creará el método distribuirBarcos que reciba como parámetro los tableros, el código de jugador y el tipo de jugador (humano u ordenador).No devolverá nada.
  - El método distribuirBarcos será llamado para cada uno de los jugadores para distribuir sus barcos.

- Desde este método se llamará a los métodos seleccionarBarco, seleccionarFila, seleccionarColumna, seleccionarOrientacion, situarBarco explicados a continuación.

Métodos *seleccionarBarco, seleccionarFila, seleccionarColumna, seleccionarOrientacion*

- Para colocar un barco habrá que seleccionar el tipo de barco a colocar, las coordenadas de la primera posición (fila y columna) y la orientación del barco (vertical u horizontal), para indicar de alguna forma cómo se colocarán las posiciones del barco a partir de la primera (arriba/abajo o izquierda/derecha).
- Para seleccionar el tipo de barco, se creará el método seleccionarBarco que reciba como parámetro los tipos de barcos que se pueden seleccionar, el código de jugador y el tipo de jugador (humano u ordenador). Devolverá un dato entero indicando el tipo de barco seleccionado por el usuario.
- Para seleccionar la fila, se creará el método seleccionarFila que reciba como parámetro el código de jugador y el tipo de jugador (humano u ordenador). Devolverá un dato entero indicando la fila seleccionada por el usuario.
- Para seleccionar la columna, se creará el método seleccionarColumna que reciba como parámetro el código de jugador y el tipo de jugador (humano u ordenador). Devolverá un dato entero indicando la columna seleccionada por el usuario.
- Para seleccionar la orientación, se creará el método seleccionarOrientacion que reciba como parámetro el código de jugador y el tipo de jugador (humano u ordenador). Devolverá un dato entero indicando la orientación seleccionada por el usuario.

Método *situarBarco*

- Una vez obtenidos los datos anteriores, se sitúa el barco en el tablero. En la colocación de los barcos habrá que tener en cuenta que:
  - El barco completo debe poder ser colocado dentro del tablero.
  - Un barco no puede situarse de manera que quede adyacente a otro barco colocado anteriormente.
- Para ello, se creará el método situarBarco que reciba como parámetro el tablero sobre el que se quiere situar el barco, el tipo de barco, la fila, la columna y la orientación. Devolverá un dato de tipo boolean, que indicará si la situación del barco es válida o no. En caso de no serlo, se ejecutarán de nuevo los métodos explicados arriba para indicar valores adecuados de tipo de barco, fila, columna y orientación.
- Según se ha indicado antes, para determinar si la situación del barco es correcta, habrá que comprobar que todas sus partes quedan dentro del tablero y no se solapan con ningún otro barco.

A continuación, se elegirá aleatoriamente a uno de los dos jugadores, que será el que realice el primer disparo.

Método *disparar*

- Para realizar el disparo, el jugador deberá elegir una fila (1-20) y una columna (A-Z). Los datos serán introducidos por teclado por los jugadores humanos; en el caso de jugadores controlados por el ordenador, la fila y la columna serán seleccionadas aleatoriamente. Se comprobará en la Zona Propia del rival si en dicha posición hay un barco o sólo agua:
  - Si el disparo cae en el agua, se indicará "¡AGUA!" y se realizarán los cambios oportunos en las zonas del tablero correspondientes de cada uno de los jugadores (Zona Enemiga del jugador atacante y Zona Propia del otro jugador), marcando las casillas como disparo caído en el agua con el símbolo '.'. La puntuación del jugador atacante disminuirá 1 punto.
  - Si el disparo cae en un barco, se indicará "¡TOCADO!" y se realizarán los cambios oportunos en las zonas del tablero correspondientes de cada uno de los jugadores (Zona Enemiga del jugador atacante y Zona Propia del otro jugador), marcando las casillas como parte del barco dañada con el símbolo 'T'. La puntuación del jugador atacante se incrementará 10 puntos.

- Si el barco ha sido hundido, se indicará "¡TOCADO y HUNDIDO!" y se realizarán los cambios oportunos en las zonas del tablero correspondientes de cada uno de los jugadores (Zona Enemiga del jugador atacante y Zona Propia del otro jugador), marcando las casillas que ocupaba el barco con el símbolo '@'. La puntuación del jugador atacante se incrementará 20 puntos.

- Para ello, se creará el método disparar que reciba como parámetro el tablero de juego, código de jugador, tipo de jugador (humano u ordenador) y su puntuación. Devolverá una variable de tipo boolean para controlar si es el final del juego (el disparo ha hundido el último barco del enemigo).
- Desde el método disparar se llamará al método quedanBarcos descrito a continuación.

Método *quedanBarcos*

- Se comprobará si el juego ha terminado (no quedan casillas de barco sin daños en la Zona Propia del jugador atacado):
  - Si es así, se mostrará un mensaje que indique el jugador que ha ganado y se saldrá de la aplicación.
  - En cualquier otro caso, se continuará con el juego (se mostrarán las zonas del tablero del otro jugador, se seleccionarán las coordenadas de su siguiente disparo, etc.).

4. Actividades optativas

- Modificar la "inteligencia" de los jugadores controlados por ordenador para que sigan diferentes estrategias a la hora de realizar sus disparos (también puede aplicarse a la colocación de sus barcos).
- Configuración del juego mediante parámetros que permitan variar las dimensiones del escenario o el número de barcos de cada uno de los tipos para cada uno de los jugadores.
- Almacenar en un fichero las puntuaciones de los jugadores y cargarlas al principio de cada partida (se proporcionará código de ayuda a los alumnos interesados).