

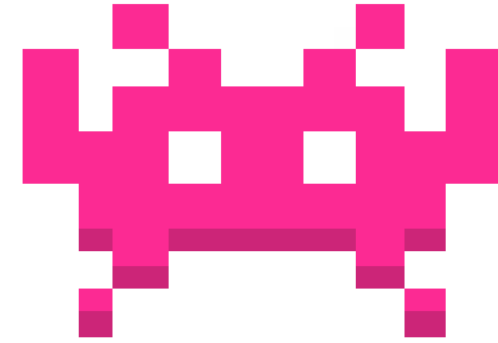
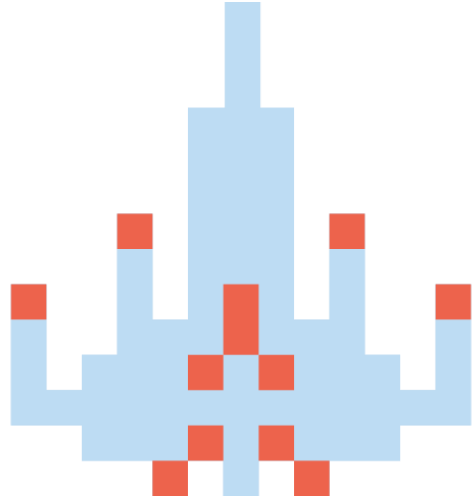


UNIVERSIDAD
POLITÉCNICA
DE MADRID

POLITÉCNICA

Verificación y Validación

CURSO 24/25



SPACE INVADERS

PRÁCTICA DE VERIFICACIÓN Y VALIDACIÓN

Objetivo

- En esta práctica, realizaréis el proceso de análisis, depuración y testing del código del juego “Space Invaders”.
- El proceso se divide en 5 niveles, tal y como sería en un videojuego.
- Los niveles son incrementales, es decir, es necesario haber superado el anterior para poder realizar el siguiente

Reparto de niveles

Bloque 1 Fecha de entrega: 15 de noviembre	Nivel 1. Análisis dinámico – Pruebas de caja negra
	Nivel 2. Análisis dinámico – Pruebas de caja blanca
Bloque 2 Fecha de entrega: 13 de diciembre	Nivel 3. Depuración de código
	Nivel 4. Pruebas de integración
	Nivel 5. Pruebas de sistema

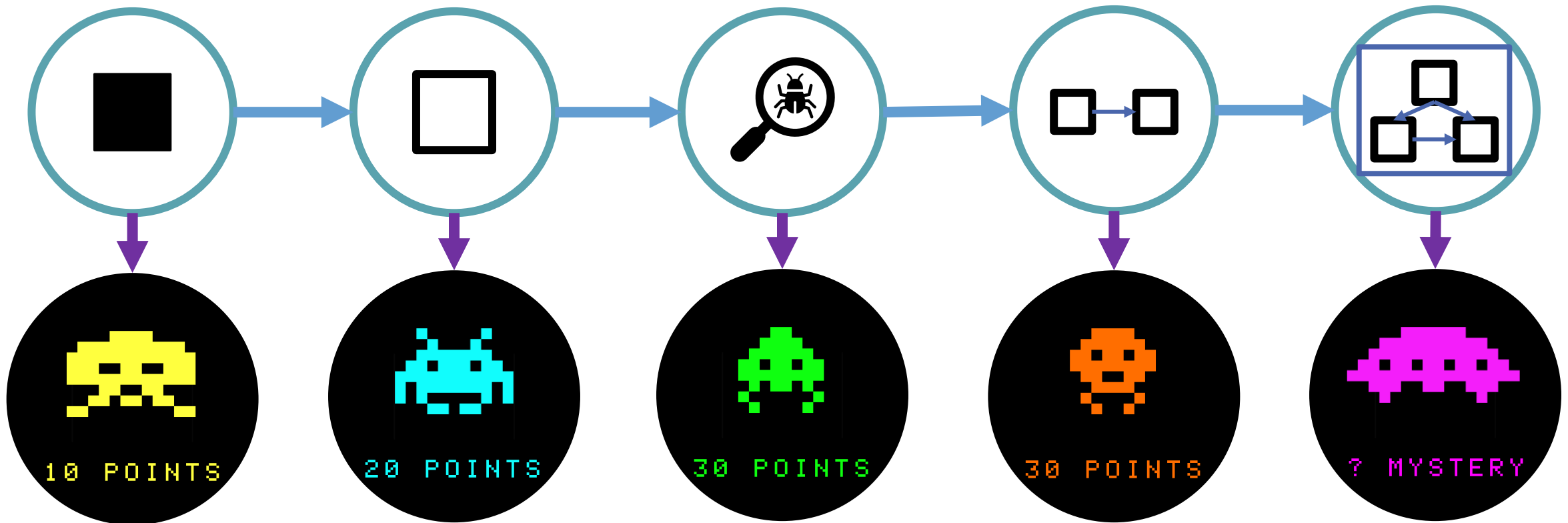


Objetivo

- Al final de la práctica, el juego debería funcionar correctamente.
- Los fallos de código deben ser detectados y corregidos
- Se deben realizar pruebas no sólo que permitan encontrar los errores, si no también verificar su correcto funcionamiento
- La superación de cada nivel otorgará una insignia en Moodle

Recompensas



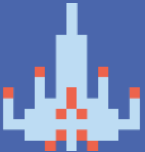

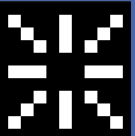
- Para los 5 grupos que obtengan mejor calificación en cada uno de los niveles, se entregará una recompensa especial



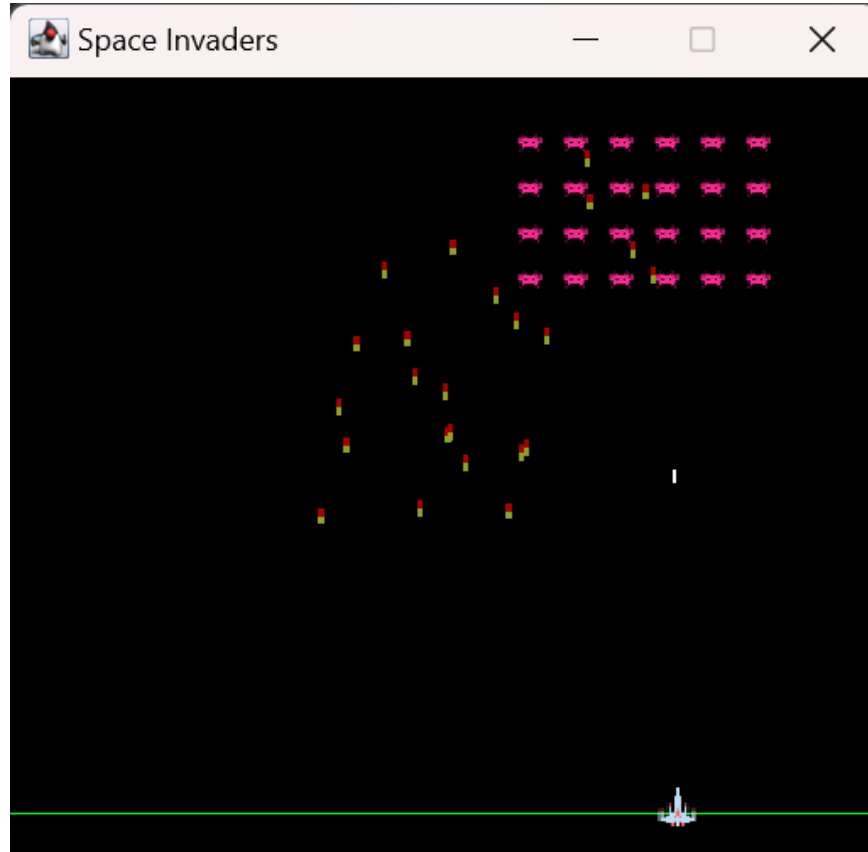
Instrucciones generales

- **Los grupos serán de, como máximo, 5 personas.** Se pueden hacer grupos más pequeños, pero no se modificará la carga de trabajo para adaptarse a esta circunstancia.
- **La práctica se realizará en Java**, utilizando Maven para la construcción. Se aportará un fichero pom.xml para la generación del entorno. Este fichero debe usarse para asegurar la correcta ejecución del código y de las pruebas, especialmente a la hora de la corrección.
- **Los contenidos de la práctica se entregarán en un único fichero.zip**, donde se incluirán todos los ficheros desarrollados y necesarios para su corrección

Contexto de la práctica

Icono	Nombre	Descripción
	Alien	Los alienígenas son los enemigos del juego. Su objetivo es destruir al jugador mediante disparos. Si los alienígenas llegan hasta la nave, ganarán el juego.
	Bombas	Los alienígenas pueden lanzar bombas al jugador para intentar derrotarlo. Si le alcanzan, el jugador es eliminado.
	Jugador	El jugador se representa mediante una nave espacial. Su objetivo es disparar y destruir a todos los alienígenas al tiempo que evita que le alcancen las bombas.
	Disparos	La nave espacial puede disparar proyectiles para destruir a los alienígenas.
	Explosión	Cuando un alienígena o un jugador es alcanzado, se produce una explosión que indica que ha sido eliminado.

Contexto del juego



Objetivo del jugador: destruir todos los alienígenas antes de que destruyan la nave o lleguen al suelo

La nave puede moverse sólo de izquierda a derecha y disparar un proyectil cada vez

Los alienígenas se mueven de un lado a otro y hacia abajo

Código de la práctica

Directorio	Fichero	Descripción
\main		
	Board.java	Crea la pantalla de juego, la configuración y coloca los elementos
	Commons.java	Indica las constantes del juego: tamaños, posiciones, etc.
	Main.java	Fichero main del juego (Se aporta para el segundo bloque)
\sprites		
	Alien.java	Construcción y métodos de los alienígenas y sus bombas
	Player.java	Construcción y métodos del jugador
	Shot.java	Construcción y métodos de los disparos
	Sprite.java	Clase de la que heredan las clases Alien, Player y Shot. Establece métodos comunes usados por todos los elementos del juego, como los “getters” y “setters” de sus posiciones en pantalla.

En total, hay 3 ficheros afectados por defectos de código, habiendo un total de 10 defectos de código repartidos por los diferentes ficheros

Bloque I: Niveles 1 y 2

- El objetivo de este primer bloque es realizar el análisis dinámico del código del juego
- **Nivel 1 - Pruebas de caja negra:** El objetivo de esta fase es analizar, usando las técnicas aprendidas, el código del juego para verificar su funcionamiento correcto y detectar posibles fallos
- **Nivel 2 - Pruebas de caja blanca:** El objetivo de esta fase es probar, usando esta aproximación, los mismos métodos probados mediante el análisis de caja negra.

Nivel 1- Pruebas de caja negra

- El este nivel se recibe tanto el código del juego como la documentación.
- En base a la documentación, se debe decidir cuál es la mejor estrategia para probar el método en cuestión.
- Una vez escogida la estrategia, se deben diseñar los casos de prueba e implementarlos usando JUnit
- Estos casos de prueba deben servir no sólo para encontrar defectos en el código, si no para asegurar también su correcto funcionamiento.

Nivel 1- Pruebas de caja negra

Se entregará:

- **Breve documento** donde se indique qué métodos han sido probados y cuál ha sido la estrategia escogida para probarlos. Las decisiones tomadas deben estar debidamente justificadas.
- **El conjunto de casos de prueba** diseñados por cada método probado, indicándose que entradas se han probado y qué salidas se esperan. Por ejemplo, se puede emplear una tabla Excel.
- Las **implementaciones** en Junit5 de los casos de prueba definidos.
- Los **resultados obtenidos** tras la finalización del nivel, es decir, el conjunto de fasos de prueba superados, y los NO superados, que pasarán a fase de depuración

Nivel 2- Pruebas de caja blanca

- En este nivel, se usará el código fuente del juego.
- Se deben probar aquellos métodos que hayan sido probados también con pruebas de caja negra.
- Este análisis permitirá corroborar los defectos encontrados y descubrir nuevos defectos, así como verificar el correcto funcionamiento del código.
- Para cada método estudiado, se realizará su grafo correspondiente, indicándose su complejidad ciclomática y el conjunto de caminos independientes.

Nivel 2 – Pruebas de caja blanca

Se entregará:

- **Un breve documento** donde se presenten los grafos de ejecución de cada uno de los métodos, su complejidad ciclomática y su conjunto de caminos independientes.
- El **conjunto de casos de prueba** diseñados para cada método, indicándose qué entradas se han probado y que salidas se espera obtener.
- Las **implementaciones** en JUnit5 de los casos de prueba definidos.
- Los **resultados obtenidos** a la finalización de esta fase, es decir, qué casos de prueba no han sido superados y pasarán a depuración, y cuáles sí han sido superados.