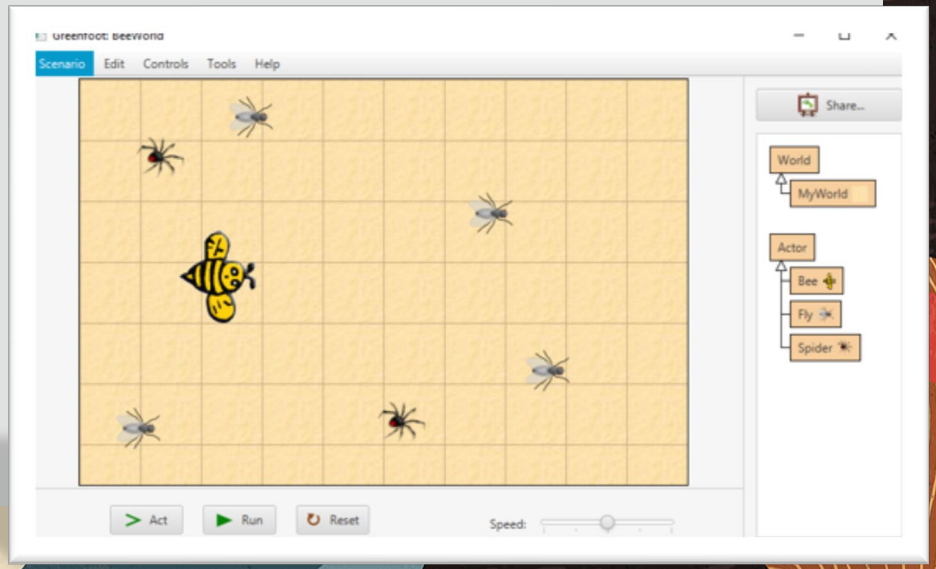

ORACLE
Academy

Creación de Programas Java con Greenfoot

Lección 5

Desarrollo y prueba de una aplicación



ORACLE
Academy

Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Demostrar las estrategias de las pruebas del programa
  - Reconocer las fases para el desarrollo de una aplicación de software

Estrategias de las pruebas del programa

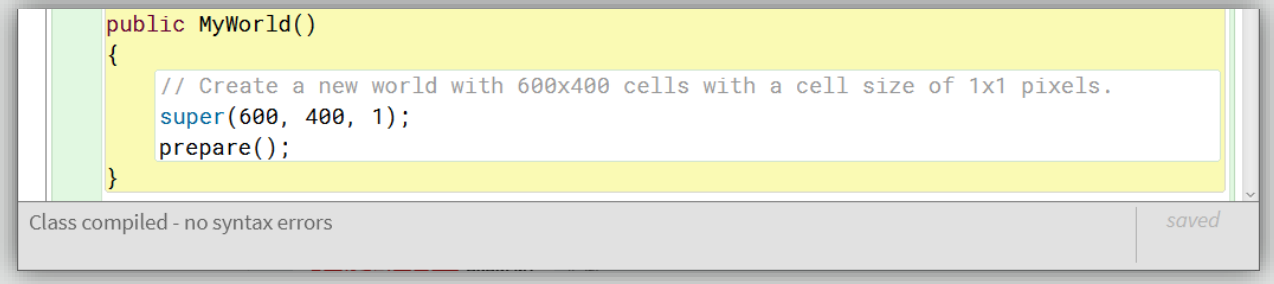
- Un programador realiza pruebas de un programa muchas veces durante el curso de su desarrollo para garantizar que funciona correctamente
- Estrategias de las pruebas del programa:
  - Realizar pruebas frecuentemente después de que se escriba cada método, o una secuencia de métodos
  - Si aparece algún error, corregirlo
  - Ejecutar el programa para observar cómo los métodos hacen que los objetos se muevan
  - Continuar para agregar métodos y ajustarlos según sea necesario

Compilación y depuración

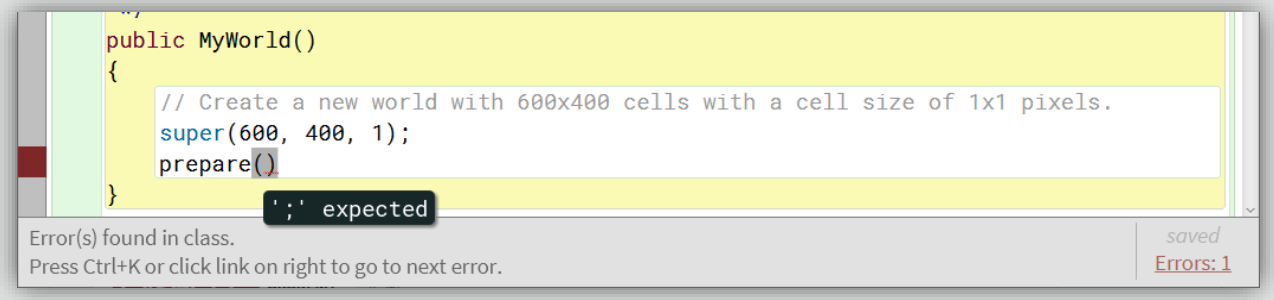
- Cada carácter del código fuente es importante
- Un carácter que falte o incorrecto podría provocar que el programa fallará
- En Greenfoot, la compilación resalta los errores de sintaxis y lo que es necesario para corregirlos
- Esto le ayudará a desarrollar buenas técnicas de programación

Pasos para depurar el programa

- Si no hay ningún error, aparece el mensaje "Class compiled – no syntax errors"



- Si hay errores, se resalta la sintaxis incorrecta y un mensaje intenta explicar el error

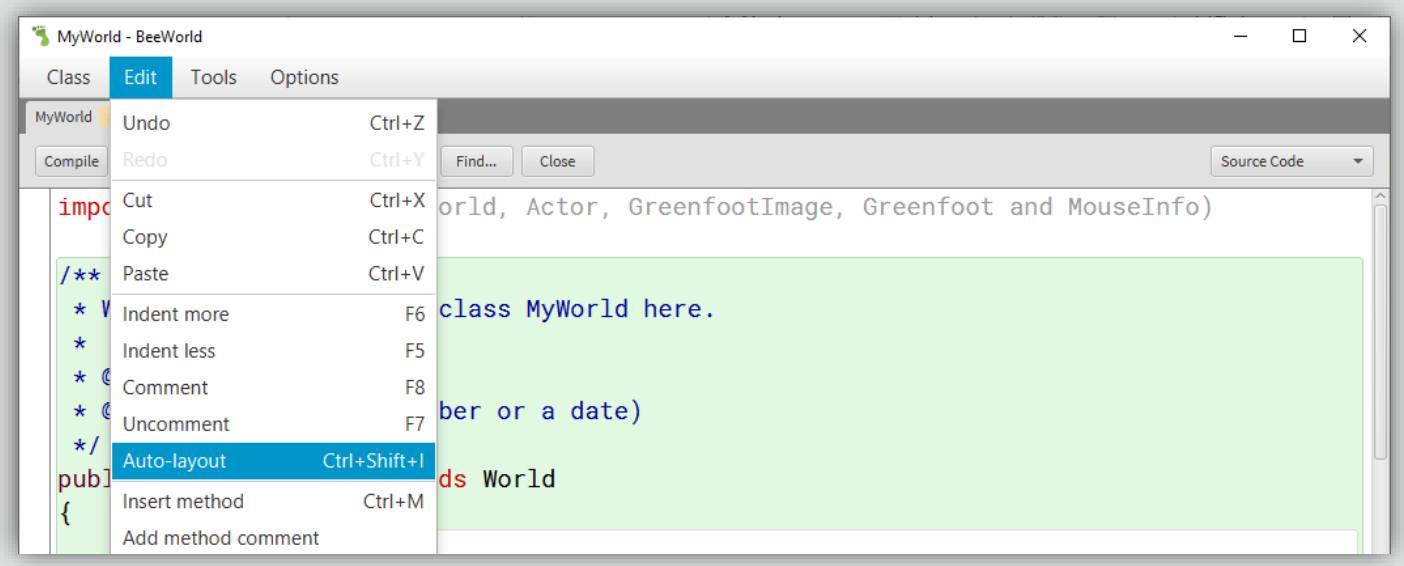


Claves para reconocer los errores de sintaxis de Java

1	Localice el principio y el final de un método.
2	Asegúrese de que existen los corchetes de inicio { y fin }.
3	Asegúrese de que existen los paréntesis de apertura (y de cierre).
4	Asegúrese de todas las líneas de código terminan con un punto y coma.
5	Asegúrese de que los nombres de clases están correctamente escritos y el uso de mayúsculas y minúsculas es el adecuado.
6	Revise la notación de todos los puntos (por ejemplo,System.out.println).
7	Asegúrese de que los caracteres de aspecto similar son correctos (como el número 1 frente a la letra i).
8	Asegúrese de que todas las comillas de una cadena son dobles " y no simples '.

Auto-Layout

- Una función útil en el editor de código de Greenfoot es la función Auto-Layout
- Observará que esta función estructura automáticamente el código y es una herramienta muy útil para buscar dónde faltan paréntesis



Fases para desarrollar una aplicación

- Analizar el problema que se debe resolver o la tarea que se va a realizar
- Diseñar la solución, que por lo general, en Greenfoot es un juego
- Desarrollar el juego de Greenfoot
- Probar el juego para asegurarse de que funciona y cumple con los requisitos de análisis y diseño
- El desarrollo de un juego en Greenfoot sigue los mismos pasos que el desarrollo de una aplicación de software

Fase de análisis

- En la fase de análisis, determine el problema que resolverá el juego, o la tarea que realizará, utilizando el análisis orientado a objetos

En el análisis orientado a objetos, los programadores de Java analizan un problema y, a continuación, crean objetos para generar un sistema, o más concretamente, para resolver el problema.

Tareas de la fase de análisis

- Identificar un problema que resolver
- Escribir una breve declaración de ámbito que indique el tipo de solución (juego) que va a resolver el problema
- Recopilar los requisitos del público objetivo
- Estas son las personas que más probablemente jueguen al juego
- Identificar y describir los objetos en el juego
  - Objetos físicos (coche, persona, árbol)
  - Objetos conceptuales ("no físicos") (temporizador que cuente el tiempo que queda en el juego)
  - Atributos de todos los objetos, como el color, tamaño, el nombre y la forma
  - Operaciones que realizan los objetos (moverse, girar, comerse a otros objetos)



Ejemplo de análisis:

Elemento de análisis	Description
Dominio de problemas	Deseo crear un juego para enseñar a los estudiantes cómo controlar un objeto Bee con las teclas de cursor.
Requisitos del jugador	Debería ser fácil de jugar para niños de todas las edades. Es necesario que el jugador tenga un teclado.
de datos	1 objeto Bee que cazará objetos Fly. Varios objetos Fly. 1 Objeto Spider que caza objetos Fly y el objeto Bee. 1 World con un fondo de color claro.
Operaciones de objetos	Bee: Mover, girar y cazar objetos Fly. Fly: Desplazarse por la pantalla de forma aleatoria. Recuento de vida: Cuenta atrás de 1 en 1 a partir de 3 cada vez que el objeto Spider captura el objeto Bee. Antecedentes: No hacer nada.

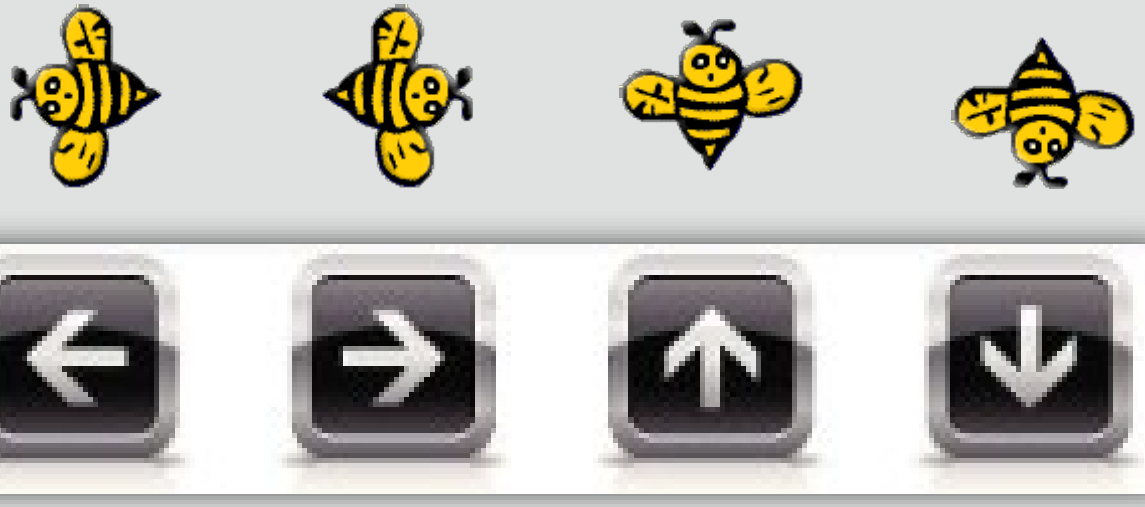
Condiciones previas y posteriores al análisis

- Capture información para apoyar las pruebas de:

Elemento para probar	Ejemplo
Condiciones previas y posteriores al juego.	Valores iniciados por variables frente a valores finales después de la ejecución del programa.
Tiempos de ejecución previstos y ratios de ejecución de comparación dadas una serie de condiciones.	Los ratios de ejecución pueden variar según la variación del tamaño de memoria del ordenador.
Los resultados esperados para los recuentos de ejecución de sentencia.	Un contador en bucle de tres producirá tres nuevas variables.
Representaciones y limitaciones numéricas.	Valor máximo de un entero.

Fase de diseño

- La solución que diseñe tendrá la forma de un juego de Greenfoot al que su público objetivo pueda jugar
- Diseñe su juego en un guión de texto que planifique los algoritmos, o métodos, que los objetos realizarán en respuesta a los comandos del teclado o unos clics del ratón



Ejemplo de guión de texto

- El guión de texto en la siguiente diapositiva describe un juego sencillo donde controla a una abeja para cazar moscas mientras evita a una araña
- La araña también cazará moscas
- Obtendrá puntos por cada mosca capturada y perderá una vida cada vez que una araña cace a la abeja
- El juego termina cuando se quede sin vidas

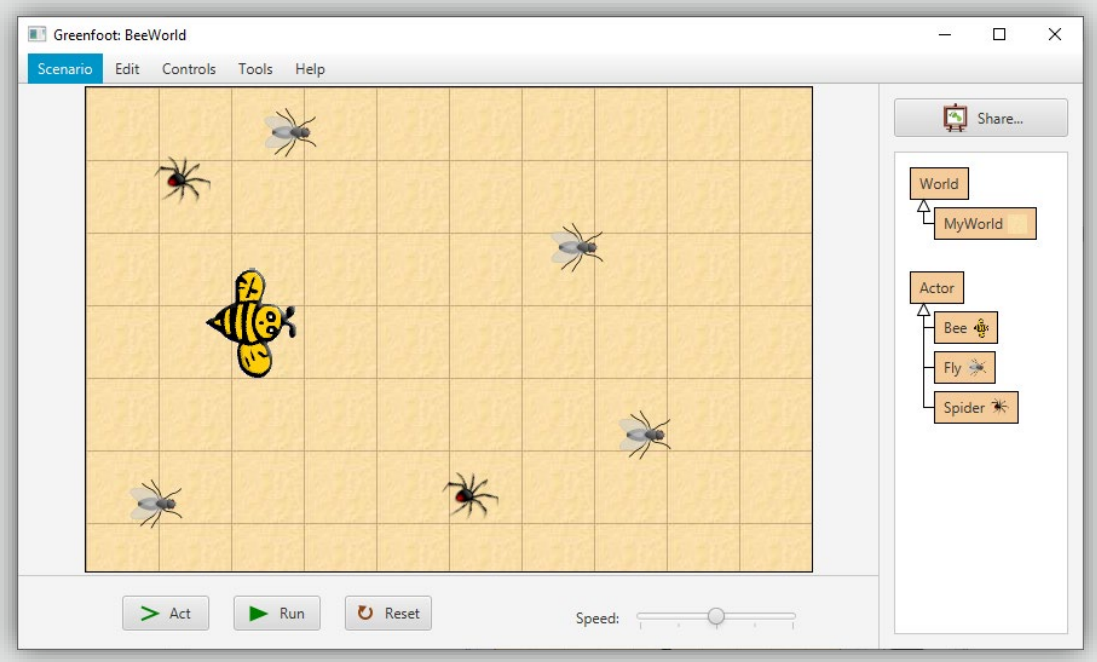


Ejemplo de guión de texto

- Cuando se hace clic en el botón Run, el objeto Bee se moverá continuamente hacia adelante
- El jugador utiliza las teclas de flecha del teclado para controlar los movimientos hacia la izquierda y derecha del objeto Bee
- Cuando el objeto Bee está en la misma cuadrícula que un objeto Fly que se mueve aleatoriamente, se captura, se elimina del juego y se agrega otro objeto Bee
- Un objeto Spider se moverá aleatoriamente por toda la pantalla
- Si el objeto Spider captura un objeto Bee, el usuario pierde una vida
- Si captura un objeto Fly, este se elimina del juego
- El juego termina cuando el usuario no tenga ninguna vida

Fase de desarrollo

- Después de finalizar el guión, desarrolle su juego en Greenfoot
- Consulte el guión para determinar los métodos que necesita programar



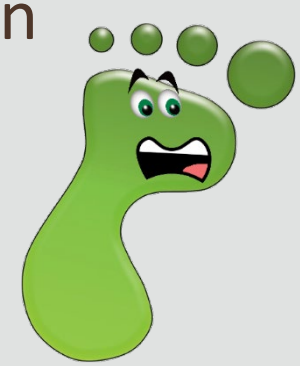
Fase de prueba

- Después de escribir una sección de código, lo compilará y, a continuación, realizará la prueba haciendo clic en el botón Run en el entorno
- Observe el juego y revise el código según sea necesario
- Con fines de control de calidad:
  - Haga que otras personas prueben el juego y le proporcionen comentarios
  - Busque personas que se ajusten al público objetivo del juego
- Escriba planes de prueba que:
  - Examinen las condiciones anteriores y posteriores
  - Comparen los ratios de tiempo de ejecución y los recuentos de ejecución
  - Prueben a fondo las representaciones numéricas y sus limitaciones

Ejemplo 1 de prueba de representaciones numéricas y límites

Por ejemplo, una solución de banca necesita un redondeo preciso de los números.

- Este programa podría producir resultados incorrectos si el redondeo de un número no se ha definido en dos dígitos después de un punto decimal
- La adición de 1/2 de céntimo multiplicado por un millón de clientes podría producir un costoso error de programación



Ejemplo 2 de prueba de representaciones numéricas y límites

Por ejemplo, un constructor IF en un programa espera un valor positivo de 5 a 9 para, a continuación, agregar el valor a otra variable.

- El programa asigna incorrectamente a la variable un valor de 2
- Esto hace que el constructor IF falle y la variable que espera un cambio no condicional no obtenga la cantidad esperada, por lo que el funcionamiento de la estructura de datos será diferente
- Este es un ejemplo en el que el programa ejecutará el resultado del constructor condicional, incluso si es incorrecto

Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Errores
  - Documentación

Inténtelo

Probar y depurar código

Esta actividad necesita que empiece con el archivo de proyecto que se guardó en el tema anterior FrogFly_L2T2.

FrogFly_L2T2.zip

Instrucciones

Abra el editor de códigos para la subclase Fly.

Elimine el corchete angular al final de la definición de la clase. Revise el mensaje de error. Vuelva a agregar el corchete angular.

Escriba incorrectamente el nombre del método de turn. Por ejemplo, trun. Revise el mensaje de error. Depure el error y vuelva a compilar el código.

Cree y depure dos errores más en el código.

Summary

- En esta lección, debe haber aprendido lo siguiente:
  - Demostrar las estrategias de las pruebas del programa
  - Reconocer las fases para el desarrollo de una aplicación de software


ORACLE
Academy