

The logo for Oracle Academy. The word "ORACLE" is in a bold, orange, sans-serif font. Below it, the word "Academy" is in a smaller, dark gray, sans-serif font. The entire logo is centered on a light gray background, which is framed by dark gray horizontal bars at the top and bottom.

ORACLE

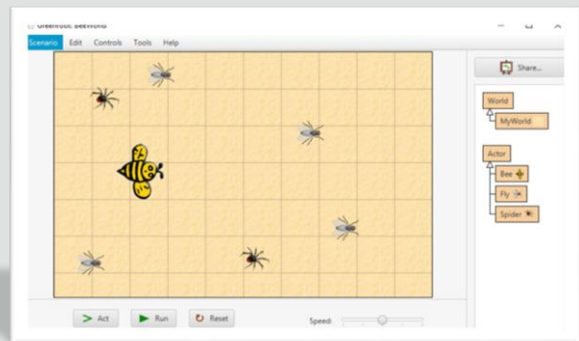
Academy

Java Fundamentals

3-4

Desenvolvendo e Testando um Aplicativo

ORACLE
Academy



Copyright © 2022, Oracle e/ou suas empresas afiliadas. Oracle, Java e MySQL são marcas comerciais registradas da Oracle Corporation e/ou de suas empresas afiliadas. Outros nomes podem ser marcas comerciais de seus respectivos proprietários.

Objetivos

- Esta lição abrange os seguintes objetivos:
 - Demonstrar estratégias de teste do programa
 - Reconhecer fases para desenvolver um aplicativo de software



Estratégias de Teste do Programa

- Um programador testa um programa muitas vezes durante o desenvolvimento para assegurar que ele esteja funcionando corretamente
- Estratégias de teste do programa:
 - Teste frequentemente após cada método ou após uma sequência de métodos ser escrita
 - Se houver erros, corrija-os
 - Execute o programa para observar como os métodos fazem com que os objetos se movam
 - Continue a adicionar métodos e faça os ajustes, conforme necessário

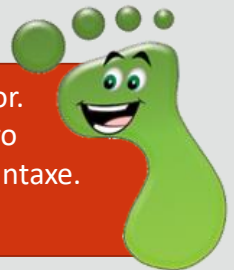
O teste é um aspecto importante no desenvolvimento do software. Você está sempre testando seu programa quando escreve o código-fonte, compila ou executa. Uma estratégia bem definida pode ajudar muito na qualidade do seu software.

Alguns aspectos do código serão testados por você, mas outros serão testados por outras pessoas. Permitir que outros usuários, principalmente aqueles para quem o software foi desenvolvido, testem o programa ajudará a remover erros e a aumentar a funcionalidade do seu software.

Compilação e Depuração

- Cada caractere no código-fonte conta
- Um caractere que esteja faltando ou que esteja incorreto pode fazer com que o programa falhe
- No Greenfoot, a compilação destaca erros de sintaxe e o que é necessário para corrigi-los
- O que ajuda a desenvolver boas técnicas de programação

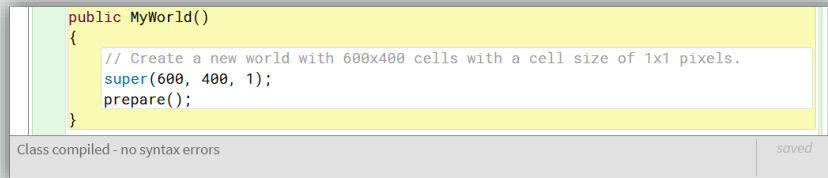
Os bugs são erros encontrados no código de um programa de computador. Para depurar um programa, o programador lê todas as mensagens de erro que o Greenfoot fornece. O programador, então, corrige esses erros na sintaxe. O teste passará, então, para a lógica no código.



Lembre-se de que uma compilação de software bem-sucedida não significa que não existam bugs; apenas indica que a sintaxe está correta.

Etapas para Depurar Seu Programa

- Se não houver erros, a mensagem "Class compiled – no syntax errors" será exibida

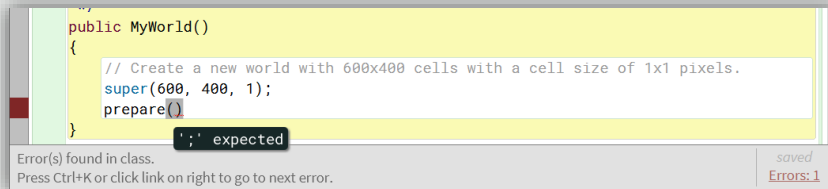


```
public MyWorld()
{
    // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
    super(600, 400, 1);
    prepare();
}
```

Class compiled - no syntax errors

saved

- Se houver erros, a sintaxe incorreta será realçada e uma mensagem tentará explicar o erro



```
public MyWorld()
{
    // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
    super(600, 400, 1);
    prepare()
}
```

';' expected

Error(s) found in class.
Press Ctrl+K or click link on right to go to next error.

saved

Errors: 1

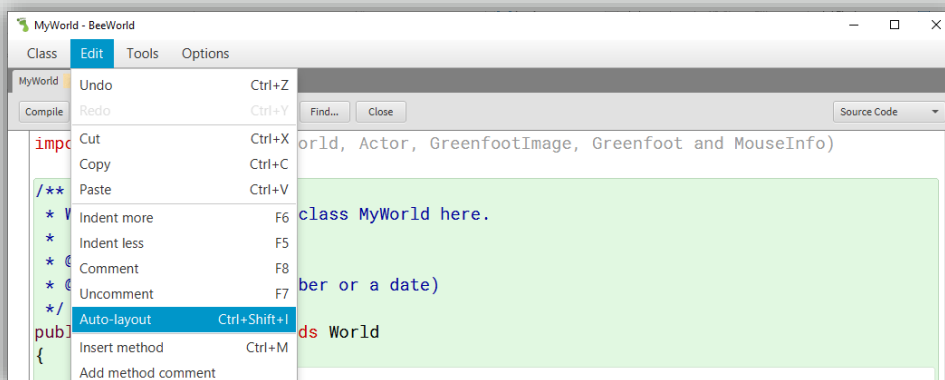
Dicas para Reconhecer Erros de Sintaxe Java

1	Localize o início e o fim de um método
2	Certifique-se de que as chaves de início ({) e de fim (}) estejam presentes
3	Certifique-se de que os parênteses de abertura e fechamento estejam presentes
4	Certifique-se de que todas as linhas de código terminem com um ponto-e-vírgula
5	Certifique-se de que os nomes das classes estejam escritos da forma correta e sejam devidamente capitalizados
6	Verifique toda a notação de pontos (ou seja, <code>System.out.println</code>)
7	Certifique-se de que caracteres de aparência semelhante estejam corretos (número 1 vs letra i)
8	Certifique-se de que as aspas nas strings sejam simples, e não duplas

O uso de um recuo de código adequado melhorará bastante a legibilidade do seu código. Assim, será mais fácil e rápido localizar erros como os mencionados acima.

Layout Automático

- Uma função de grande utilidade no code editor do Greenfoot é o recurso Layout Automático
- Como você poderá perceber, ela estruturará automaticamente seu código, sendo muito útil para localizar onde estão faltando parênteses



O layout automático recuará o código entre parênteses. Isso demonstra boas técnicas de layout de programa para tornar o código mais legível. O Greenfoot permite que você escreva todo o código em uma única linha, mas assim será mais difícil encontrar erros no código. Além disso, essa decisão fará com que a leitura do código seja uma tarefa bastante onerosa.

Fases para Desenvolver um Aplicativo

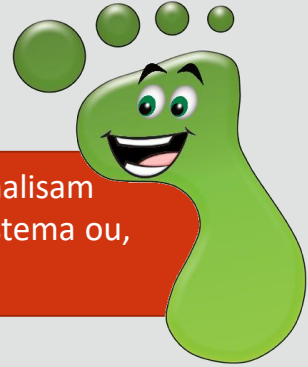
1. Analisar o problema a ser solucionado ou a tarefa a ser executada
2. Projetar a solução, que costuma ser um jogo no Greenfoot
3. Desenvolver o jogo no Greenfoot
4. Testar o jogo para assegurar que ele funcione e atenda aos requisitos de análise e projeto
5. O desenvolvimento de um jogo no Greenfoot segue as mesmas etapas que o desenvolvimento de um aplicativo de software

Planejar seu jogo antes de começar a codificá-lo economizará muito tempo. Alguns jogos simples exigirão muito pouco planejamento, mas, à medida que a complexidade do jogo aumenta, também cresce a necessidade de técnicas de planejamento adequadas.

Fase de Análise

- Na fase de análise, determine qual problema o jogo solucionará ou a tarefa que ele executará usando a análise orientada a objetos

Na análise orientada a objetos, os programadores Java analisam um problema e podem criar objetos para construir um sistema ou, mais especificamente, solucionar o problema.



A identificação dos objetos exigidos no software ajudará você a determinar o número de subclasses necessárias na classe Ator. Embora geralmente teremos um nível de classes em Ator, nos programas maiores, temos vários níveis em que existe Ator -> subclasse->subclasse em que as classes compartilham campos e métodos comuns.

Tarefas da Fase de Análise

- Identificar um problema a ser solucionado
- Escrever uma instrução resumida do escopo que indica o tipo da solução (jogo) que resolverá o problema
- Reunir os requisitos do público-alvo
- São essas pessoas que mais provavelmente jogarão seu jogo
- Identificar e descrever os objetos no jogo
 - Objetos físicos (carro, pessoa, árvore)
 - Objetos conceituais ("não físicos") (cronômetro que conta o tempo restante no jogo)
 - Atributos de todos os objetos, como cor, tamanho, nome e forma
 - Operações que os objetos executam (mover, girar, comer outros objetos)



A coleta das informações necessárias ajudará você a planejar uma solução.

Exemplo de Análise

Item de Análise	Descrição
Domínio do problema	Quero criar um jogo para ensinar os alunos a controlarem uma Abelha com as teclas do cursor
Requisitos dos jogadores de um jogo	Precisa ser um jogo fácil para crianças de todas as idades jogarem. O jogador precisa ter um teclado
Objetos	1 objeto Abelha capturará moscas Vários objetos Mosca 1 objeto Aranha que capturará moscas e a Abelha 1 Mundo com um plano de fundo colorido
Operações com objetos	Abelha: move-se, gira e captura moscas Mosca: move-se aleatoriamente ao redor da tela Contagem de vidas: diminui a contagem em uma unidade a partir de 3 toda vez que a Abelha for capturada por uma Aranha Plano de fundo: não faz nada



A definição das ações de um objeto dará a você a base dos métodos e dos campos necessários nas suas classes.

Pré-Condições e Pós-Condições de Análise

- Informações coletadas para suportar o teste de:

Item a ser Testado	Exemplo
Pré-condições e pós-condições do jogo	Valores inicializados da variável em comparação com valores finais após a execução do programa
Run-times antecipados e taxas de execução comparativas dado um conjunto de condições	As taxas de execução podem variar de acordo com a variação do tamanho da memória do computador
Resultados esperados das contagens de execução das instruções	Um contador de loops igual a três produzirá três novas variáveis
Limitações e representações numéricas	O valor máximo de um número inteiro

O teste pode ser planejado antes do início de qualquer codificação. Dessa forma, os programadores podem pensar no que será testado quando começarem a codificar uma solução.

Fase de Projeto

- A solução que você projetar estará no formato de um jogo do Greenfoot que seu público-lavo poderá reproduzir
- Projete o seu jogo em um storyboard textual que planeje os algoritmos, ou os métodos, que os objetos executarão em resposta aos comandos de teclado ou aos cliques do mouse



Um projeto adequado permite que você pense em como todos os seus objetos agirão e interagirão. Quando estamos gravando um código que não segue um projeto, é fácil nos atermos somente ao problema atual e não termos noção do cenário global, o que pode levar a soluções com codificações insatisfatórias.

Exemplo de Storyboard Textual

- O storyboard textual no próximo slide descreve um jogo simples em que você controla uma Abelha para tentar capturar Moscas e, ao mesmo tempo, tentar fugir de uma Aranha
- A aranha também pegará moscas
- Você ganhará pontos para cada Mosca capturada e perderá uma vida cada vez que uma Aranha capturar a Abelha
- O jogo termina quando você perde todas as vidas



Exemplo de Storyboard Textual

- Quando você clica no botão Run, a Abelha move-se continuamente para frente
- O jogador usa as teclas de seta no teclado para controlar os movimentos da abelha para a esquerda e para a direita
- Quando a Abelha estiver no mesmo quadrado que uma Mosca voando de forma aleatória, ela será pega e retirada do jogo. Outra Mosca será adicionada



Exemplo de Storyboard Textual

- Uma Aranha se moverá de maneira aleatória ao redor da tela
- Se a aranha pegar uma Abelha, o usuário perderá uma vida
- Se a aranha pegar uma mosca, ela será removida do jogo
- O jogo terminará quando o usuário não tiver mais vidas restantes



Um storyboard textual está completo no momento em que você pode entregá-lo a quaisquer programadores e todos serão capazes de produzir resultados muito semelhantes entre si. Se cada um criar uma solução totalmente diferente, então isso indica que o storyboard estava incompleto.

Você pode testar seu storyboard entregando-o a três pessoas e pedindo a elas que expliquem como o jogo funciona. Se houver muitas diferenças nas explicações dadas, então seu storyboard requer informações adicionais.

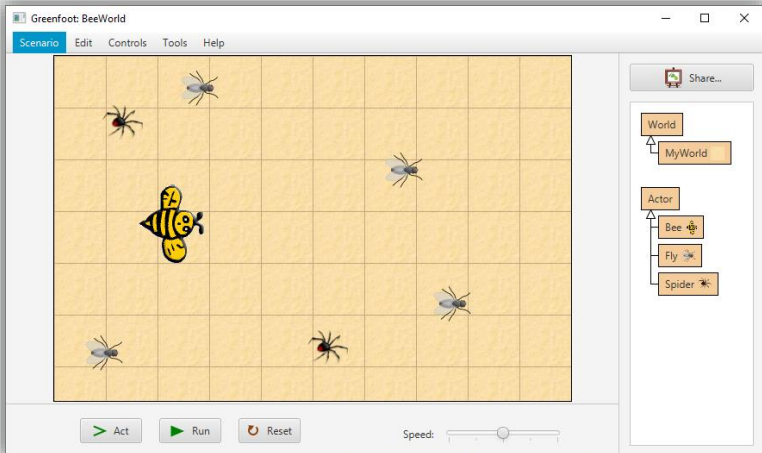
Fase de Desenvolvimento

- Depois de finalizar o storyboard, desenvolva o jogo no Greenfoot
- Consulte o storyboard para determinar os métodos necessários para programar



ORACLE
Academy

JF 3-4
Desenvolvendo e Testando um Aplicativo



Copyright © 2022, Oracle e/ou suas empresas afiliadas. Oracle, Java e MySQL são marcas comerciais registradas da Oracle Corporation e/ou de suas empresas afiliadas. Outros nomes podem ser marcas comerciais de seus respectivos proprietários.

18

Neste slide, não escrevemos um código. Apenas criamos as classes necessárias e adicionamos as instâncias dessas classes ao nosso cenário para ter uma ideia de qual será a aparência do programa.

Fase de Teste

- Depois de escrever uma seção de código, compile-o e teste-o clicando no botão Run no ambiente
- Observe o jogo e, se necessário, revise o código
- Para fins de controle de qualidade:
 - Peça a outras pessoas que testem o jogo e forneçam um feedback
 - Procure pessoas que se enquadrem no público-alvo do jogo
- Escreva planos de teste que:
 - Examinem as pré-condições e as pós-condições
 - Comparem taxas de run-time e contagens de execução
 - Testem por completo as representações numéricas e as limitações

Testar o programa em pequenos estágios permite localizar os erros mais facilmente porque você terá uma ideia mais precisa de onde eles estão. Se você escreveu todo o programa antes de testá-lo, será bem mais trabalhoso descobrir onde estão esses erros.

Testando Representações Numéricas e Limites - Exemplo 1

- Por exemplo, uma solução bancária requer o arredondamento preciso dos números
- Este programa pode produzir resultados incorretos se o arredondamento de um número não foi definido para dois dígitos depois da vírgula decimal
- A adição de metade um centavo multiplicado por um milhão de clientes pode produzir um erro de programação de custo elevado



Testando Representações Numéricas e Limites - Exemplo 2

- Por exemplo, se uma construção IF em um programa esperar um valor positivo de 5 a 9, some esse valor a outra variável
 - O programa alimenta incorretamente a variável com um valor 2
 - Isso faz com que a construção IF falhe e a variável que está esperando uma alteração condicional não obtenha o valor esperado Dessa forma, a operação da estrutura de dados será diferente.
 - Esse é um exemplo de onde o programa executará o resultado da construção condicional, mesmo que esteja incorreto

Terminologia

- Estes são os principais termos usados nesta lição:
 - Bugs
 - Documentação

Resumo

- Nesta lição, você deverá ter aprendido a:
 - Demonstrar estratégias de teste do programa
 - Reconhecer fases para desenvolver um aplicativo de software



