Engenharia de Software

Anhanguera


AVALIE
SUA PROFISSÃO
QUANDO APARECER EM SEU PORTAL UMA AVALIAÇÃO SOBRE SEU CURSO, RESPONDA:
NOTAS
9 OU 10
SIGNIFICA QUE VOCÊ INDICA
NOTAS
7 OU 8
SIGNIFICA QUE VOCÊ NÃO INDICA
Anhanguera

Anhanguera

CONCEITOS DE TESTES DE SOFTWARE

O QUE SÃO TESTES DE SOFTWARE?

Os testes são processos que representam uma sequência de ações executadas com o objetivo de encontrar problemas no software, o que aumenta a percepção da qualidade geral dele e garante que o usuário final tenha um produto que atenda às suas necessidades (PINHEIRO, 2015).

Já vai longe o tempo em que um sistema computacional tinha sua execução restrita apenas a um computador de mesa e servia a propósitos típicos do meio corporativo, como controle de vendas e de estoques. Atualmente, há uma grandevariedade de equipamentos que dependem de sistemas para executarem suas funções de modo muito mais flexível e confiável do que seria sem uma aplicação computacional. Exemplos não faltam e nossos carros e televisores são apenas dois deles. Há programas sendo executados em todos os lugares e controlando dispositivos indispensáveis em nosso cotidiano. Mas o que aconteceria se esses programas falhassem?

Para que essa pergunta não precise ser respondida da pior forma, a Engenharia de Software desenvolveu mecanismos para que os produtos de software – tantos os comuns quanto os que dão “inteligência” aos nossos equipamentos – criados pelos desenvolvedores passassem por processos que atestassem sua aptidão para executar suas funções de forma adequada e com elevados níveis de qualidade. Esses processos incluem a verificação, a validação e os testes de software e, se pormeio de uma análise menos atenta eles podem parecer idênticos, uma conceituação apropriada servirá para esclarecer seus conceitos e diferenciá-los.

GERENCIAMENTO DA QUALIDADE DO SOFTWARE

No entanto, antes de individualizarmos esses termos, vale a pena posicioná-los emum contexto mais amplo, relacionado à qualidade de um produto, e conhecido porGerenciamento da Qualidade do Software (ou Software Quality Management).Nesse contexto, a verificação e a validação são colocadas como ações intimamente relacionadas, destinadas à averiguação da conformidade do produto com as necessidades do cliente e comumente referenciadas em conjunto, sob a sigla V&V.

De acordo com uma importante publicação da área de Engenharia de Software(IEEE, 2014), o objetivo da V&V é ajudar a organização que desenvolve software a incorporar qualidade ao sistema durante o ciclo de vida, por meio de avaliações objetivas de produtos e de processos. Tais avaliações demonstram se os requisitos são corretos, completos, precisos, consistentes e testáveis. Os processos de V&V determinam se os produtos desenvolvidos em certa atividade estão em conformidade com os requisitos dela e se o produto satisfaz seu uso pretendido.

As atividades de V&V, portanto, não são aplicadas unicamente a um programa ou função, mas a qualquer artefato que seja criado como resultado de determinada etapa do ciclo de vida de um produto. Os requisitos, o projeto e a implementação do produto são artefatos que podem (e devem) passar por verificações evalidações.
Nesse sentido, a IEEE (2014) estabelece que a verificação é uma tentativa degarantir a correta construção do produto, com vistas a atender as especificaçõesimpostas aos produtos de saída em atividades anteriores. Já a validação é uma tentativa de garantir que o produto certo seja construído, ou seja, que ele atenda a sua finalidade específica.

Nesse sentido, a IEEE (2014) estabelece que a verificação é uma tentativa degarantir a correta construção do produto, com vistas a atender as especificaçõesimpostas aos produtos de saída em atividades anteriores. Já a validação é umatentativa de garantir que o produto certo seja construído, ou seja, que ele atenda asua finalidade específica.

Embora sejam expressões parecidas e estejam inseridas em um contexto único, verificação e validação não são a mesma coisa. Santos e Oliveira (2017) resumem assim os termos: verificação refere-se à garantia das especificações do software em uma determinada fase do desenvolvimento, enquanto a validação se refere àgarantia do produto de software como um todo. A validação é uma fase mais geral, na qual o produto criado é confrontado com as expectativas do cliente.

Observadas sob uma perspectiva teórica, expressões como “incorporar qualidade ao sistema” e “verificar se uma atividade está em conformidade com os requisitos”podem transmitir uma falsa sensação de simplicidade procedimental. Entretanto, a aplicação de procedimentos de verificação e validação requerem planejamentocuidadoso e precisão na execução. O objetivo do planejamento de V&V é garantir que cada recurso, função e responsabilidade sejam claramente atribuídos.

Nessa fase de planejamento, devem ser especificados os recursos, suas funções eatividades, bem como as técnicas e ferramentas a serem usadas. A compreensão dos diferentes propósitos de cada atividade de V&V ajuda no planejamento cuidadoso das técnicas e dos recursos necessários para cumprir seus propósitos. O planejamento também deve abordar a gestão, a comunicação, as políticas e os procedimentos das atividades de V&V.

Conforme mencionamos, a verificação e a validação estão incluídas em um escopo mais abrangente e estão vinculadas à Garantia da Qualidade do Software (SoftwareQuality Assurance ou SQA). Na visão de Pressman e Maxim (2016), a verificação e avalidação incluem grande variedade de atividades de SQA, quais sejam as revisões técnicas, auditorias de qualidade, monitoramento do desempenho, simulação, estudo de viabilidade, teste de usabilidade e testes de aceitação e de instalação. Osautores complementam que, embora a aplicação de teste tenha um papel extremamente importante em V&V, muitas outras atividades são necessárias.

À propósito, a menção aos testes vem a calhar. Como podemos posicioná-lo nesse contexto? O teste é a última frente de preservação da qualidade, mas não pode ser entendido como a garantia total de que a produção entregue pelas equipes estálivre de defeitos. O teste proporciona o último elemento a partir do qual a qualidade pode ser estimada e, de forma mais pragmática, os defeitos podem ser encontrados. A qualidade é incorporada ao software por meio da correta aplicaçãodas técnicas da Engenharia de Software e é confirmada durante o teste(PRESSMAN; MAXIM, 2016).

TESTES DE SOFTWARE

Um teste não é um procedimento isolado que pode ser concluído por um único membro da equipe. Embora seja comum tratá-lo por “teste”, sua execução depende de um conjunto de ações e procedimentos executados por vários elementos da equipe de desenvolvimento. Por isso, melhor seria chamá-lo de processo de teste, já que formalmente ele representa uma sequência de ações executadas com o objetivo de encontrar problemas no software, o que aumenta apercepção da qualidade geral dele e garante que o usuário final tenha um produto que atenda às suas necessidades (PINHEIRO, 2015).

É necessário, no entanto, destacar que o objetivo do teste não é o de garantir que um programa seja absolutamente livre de defeitos, mas o de encontrar problemas no software. Por mais que essa premissa nos soe estranha (e um pouco frustrante)ela deriva do fato de que nenhum teste é capaz de assegurar 100% de ausência dedefeitos em um sistema. Logo, se o processo de teste não revelar defeitos, há quese aprimorar o processo de forma geral. Não se pode considerar que o sistema não possui problemas se o teste não os revelar.

Mas, afinal, o que é teste de software? Como ele se efetiva? Há um plano a ser executado?

É natural que uma definição retirada de uma publicação de caráter estritamente técnico peça uma explicação que a aproxime da percepção comum que temos a respeito de teste.

A expressão “verificação dinâmica” indica que aplicar um teste significa, necessariamente, executar o programa. Além disso, essa execução deve ser baseada em entradas previamente selecionadas. Conforme estudaremos em detalhes na sequência, casos de teste são entradas fornecidas ao programa e às respectivas saídas esperadas. Na teoria, as entradas possíveis para um programa são infinitas, daí a necessidade de restringi-las em um “conjunto finito de casos deteste” escolhidos criteriosamente.

Bem, mas como os testes são feitos? Há uma ferramenta computacional que os execute? Nas seções futuras desta unidade, teremos a oportunidade de abordar esses assuntos com mais riqueza de detalhes, mas convém termos agora umarápida visão de um programa sendo testado. Para que isso seja possível, algumaspremissas devem ser apresentadas:

O programa em teste foi criado e está sendo executado no Eclipse, umimportante ambiente integrado de desenvolvimento (ou IDE – Integrated Development Environment) utilizado principalmente para criação e para testede aplicações Java.

O tipo de teste em questão é o de unidade. Por meio dele, apenas uma unidadedo programa (uma função ou uma classe, por exemplo) é testada e não oprograma todo.

A ferramenta utilizada para a efetivação do teste é o JUnit, que já se encontrainstalada nativamente no Eclipse.

Na Figura 3.1, você vê uma classe – chamada CalculoTest –, que representa umcaso de teste. Note que há variáveis com os valores 10 e 5 e uma terceira variável que contém o valor esperado para a execução da unidade que, no caso, realiza uma operação de soma.

```
1  package processos;
2  import junit.framework.TestCase;
3  public class CalculoTest extends TestCase{
4      public void testeExecutaCalculo() {
5          //Define os valores a serem calculados e testados
6          float PassaValor1 = 10;
7          float PassaValor2 = 5;
8          float RetornoEsperado = 15;
9          //Executa o método "ExecutaCalculo" da classe Calculo e
10         //armazena o resultado em uma variável
11         float RetornoFeito = Calculo.ExecutaCalculo(PassaValor1, PassaValor2);
12         //compara o valor retornado com o valor esperado
13         assertEquals (RetornoEsperado, RetornoFeito, 0);
14     }
15
16
17 }
```

A classe a ser testada para este caso de teste está descrita na Figura 3.2 e se chama Calculo.

```
1  package processos;
2  public class Calculo {
3      public void testeExecutaCalculo() {
4          public static float ExecutaCalculo (float Valor1, float Valor2) {
5              float Soma = Valor1 + Valor2;
6              return Soma;
7
8          }
9      }
10
11
12 }
```

A execução da classe CalculoTest através do JUnit retornará sucesso para o casode teste em questão. Dessa forma, uma função pode ser testada por meio dessaferramenta.

PLANO DE TESTES

Bem, como já compreendemos o conceito de teste, sua efetivação depender á apenas da iniciativa do seu desenvolvedor em executar o programa com base em alguns casos de teste, não é mesmo? Responder “sim” a essa pergunta equivale a ignorar uma etapa absolutamente indispensável neste contexto: o planejamento.

Pressman e Maxim (2016) ensinam que teste é um conjunto de atividades queprecisam ser planejadas com antecedência e executadas com base em um procedimento padrão, fato que motiva a criação de um modelo para o teste. Tal modelo, segundo os autores, deverá prever o emprego de técnicas específicas no projeto de casos de teste e no método de teste.

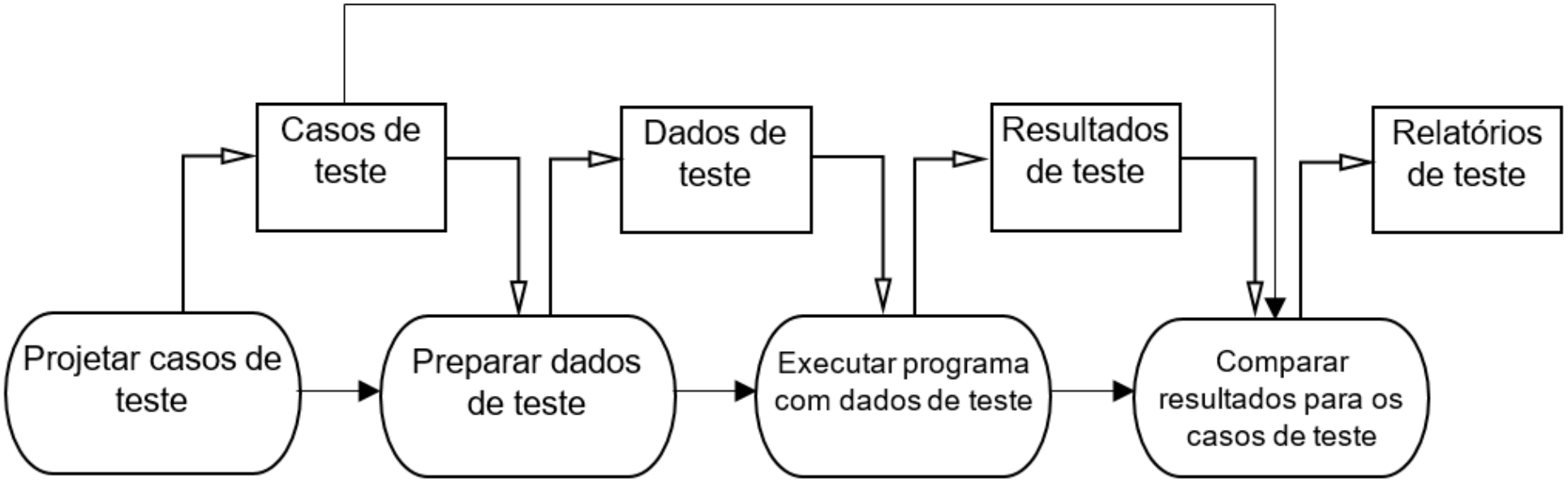
Uma definição que deve fazer parte do plano de testes é a de quem deve executá-los. Schach (2008) pondera que a realização dos testes equivale a um processo destrutivo e, como era de se esperar, um programador não desejará “destruir” seu próprio trabalho. Por isso, atribuir a atividade de teste ao mesmo time que desenvolveu o produto certamente não é uma boa ideia, já que é grande a chance de a equipe entender que deve proteger seu programa e não deve criar testes que possam revelar, de fato, problemas no código.

Outra razão para evitar a designação dos criadores do programa como seustestadores é o fato de que um terceiro poderá detectar uma falha noentendimento dos requisitos que passou despercebida ao programador e que foi implementada incorretamente. Um teste feito por outras pessoas pode aumentara chance de descoberta do problema antes que ele tenha reflexos na operação docliente.

Isto posto, é necessário mencionar que um plano de teste é normalmenteseparado em quatro grandes etapas:

Planejamento: nesta etapa deve ser definido quem executa os testes, em queperíodo, com quais recursos (ferramentas e computadores, por exemplo) equal será a técnica utilizada (técnica estrutural ou técnica funcional, porexemplo).
Projeto de casos de teste: aqui são definidos os casos de teste que serãoutilizados no processo. No próximo item, esse conceito será detalhado.
Execução do programa com os casos de teste: nesta etapa, o teste éefetivamente realizado.
Análise dos resultados: aqui se verifica se os testes retornaram resultadossatisfatórios.

A Figura 3.3 ilustra um modelo de plano de teste.



Bem, aparentemente os casos de teste são, de fato, o elemento central noprocesso de teste.

CASOS DE TESTE

Um caso de teste é o par formado por uma entrada possível a ser dada no programa e a correspondente saída esperada, também dada pelo programa e de acordo com os requisitos previamente especificados. Nesse caso, devemos entender o conceito de entrada como o conjunto de dados necessários para uma execução do programa e o de saída esperada como o resultado daquela execução ou de função específica. Com um exemplo esses conceitos serão mais bem esclarecidos: imagine que você esteja testando um programa (ou função) que promove a validação de datas inseridas pelo usuário. Um caso de teste possível seria formado pelo par (31/12/2020; válida). Ao receber a entrada 31/12/2020, afunção de validação deveria retornar “data válida”.

A boa escolha dos casos de teste é fator crítico para o sucesso da atividade. Um conjunto de casos de teste de baixa qualidade pode não exercitar partes críticas do programa e, em consequência disso, acabar não revelando defeitos no código. Oque aconteceria, por exemplo, se o responsável pelos testes usasse apenas datas válidas como entradas? No mínimo, a parte do programa que trata das datas inválidas não seria executada, o que prejudicaria a confiabilidade do processo de teste e do produto testado.

Será, entretanto, que os casos de teste apresentam apenas esse aspecto? Observe o desenvolvimento de outro exemplo: imagine que o cenário de teste agora seja achecagem da funcionalidade de login em um sistema de reserva de passagens feito por agentes de viagens. A verificação a ser feita se apoia na resposta do sistema adiversos padrões de entrada de nome de usuário e de senha. Selecionamos três deles para fins de exemplificação:

1. Checagem da resposta do sistema no caso de o agente entrar com nome deusuário e senha válidos.
2. Checagem de resposta do sistema no caso de o agente entrar com nome deusuário e/ou senha inválidos.
3.Checagem de resposta do sistema no caso de o agente pressionar a tecla Entercom o campo de nome de usuário vazio.

Para que possamos tornar mais específico nosso cenário, trataremos apenas doprimeiro item e, com base nele, apresentamos o Quadro 3.1, que descreve umcaso de teste relacionado.

Cenário	Caso de teste	Passos do teste	Dados de entrada	Resultado esperado	Resultado obtido
Checagem da funcionalidade de login.	Checagem da resposta ao se inserir nome de usuário e senha válidos.	a. Executar a aplicação. b. Informar o nome do agente. c. Informar a senha. d. Acionar o botão "Ok".	Nome do agente: Marcio Senha: 5555	O login deve ser bem-sucedido.	Login bem-sucedido.

Dessa forma, o caso de teste estará mais bem detalhado e as condições para queseja verificado estarão especificadas por completo. Note que até os passos para arealização do teste – que nos parecem tão óbvios – estão descritos nele. Fica claroque o sucesso no procedimento de testes está diretamente relacionado à boa escolha e ao bom uso dos casos de teste. Idealmente, cada conjunto de casos deteste deverá estar associado a um grande requisito diferente a ser testado. Paraque não se corra o risco de defini-los incorretamente, é necessário planejamento e bom conhecimento da aplicação. Uma boa forma de se abordar o problema é aque segue, segundo Pinheiro (2015): definir o ambiente no qual o teste será realizado, definir a entrada desse caso de teste, definir a saída esperada para cada entrada e definir os passos a serem realizados para executar os testes.

Quando um caso de teste é executado, seu resultado deve ser coletado. Podemosassumir diferentes abordagens para definir o resultado da aplicação de um caso deteste específico. A mais comum define as seguintes opções (PINHEIRO, 2015):

Passou: todos os passos do caso de teste foram executados com sucesso paratodas as entradas.

Falhou: nem todos os passos foram executados com sucesso para uma oumais entradas.

Bloqueado: o teste não pôde ser executado, pois o seu ambiente não pôde serconfigurado.

DEPURAÇÃO

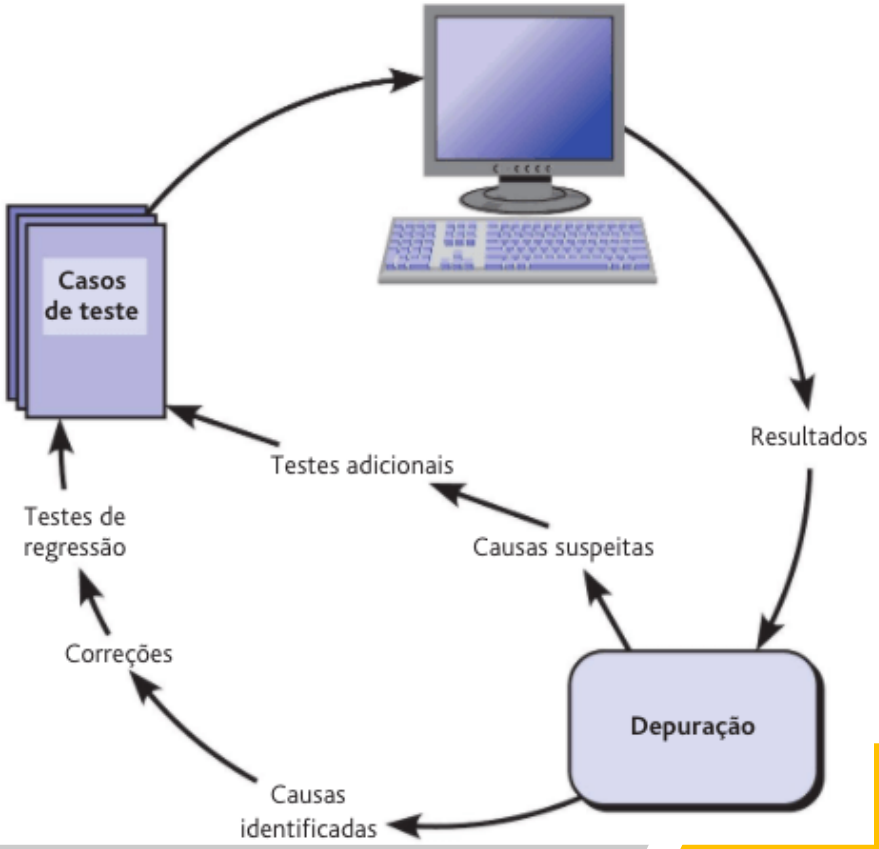
Enquanto testar significa executar o software para encontrar defeitosdesconhecidos, a depuração (ou debug) é a atividade que consiste em buscar alocalização desses defeitos no código. O fato de saber que há um problema causador de erro no programa não significa, necessariamente, que o testador sabe também em qual ou em quais linhas o problema está. Os ambientes de programação atuais oferecem recursos para depuração do programa e, duranteesse processo, o valor assumido pelas variáveis sob inspeção em cada passo doalgoritmo pode ser observado. Além disso, alguns pontos de parada da execução do programa podem ser inseridos no código. Tudo para possibilitar que o testador identifique e isole o defeito no código.

Na visão de Pressman e Maxim (2016), a depuração ocorre como consequência de um teste bem-sucedido, ou seja, quando um caso de teste descobre um defeito. Nesse caso, a depuração é o processo que encontra e remove o erro. Embora não seja um teste, ela ocorre como um desdobramento dele e começa com a execução de um caso de teste. O procedimento de depuração pode apresentar um entre dois resultados possíveis:

A causa é encontrada e corrigida e a depuração daquele problema é bem-sucedida.

A causa não é encontrada e, nesse caso, o profissional que está realizando oprocedimento tenta outro caso de teste que lhe pareça mais adequado para aquela causa.

A Figura 3.4 ilustra um procedimento de depuração:



A figura anterior nos revela o passo a passo do processo de depuração:

1. A depuração começa com a execução de um caso de teste.
2. Os resultados são avaliados e o desempenho apurado é diferente dodesempenho esperado para aquele caso de teste.
3. Quando as causas do defeito são plenamente identificadas, o problema écorrigido. Não havendo a perfeita correspondência entre o problema e a causa,procuram-se causas suspeitas.

Observe a natureza cíclica do procedimento e a colocação de “causas suspeitas”como um dos elementos do ciclo. Essa nomenclatura revela que, embora adepuração deva obedecer a um processo ordenado, ela depende bastante aindada experiência e da sensibilidade do testador. Um engenheiro de software, ao avaliar os resultados do teste, pode perceber sintomas de um problema e não asua manifestação inequívoca, o que usualmente indica que a manifestação externado problema e a sua real causa interna podem não ter nenhuma relação aparente uma com a outra.

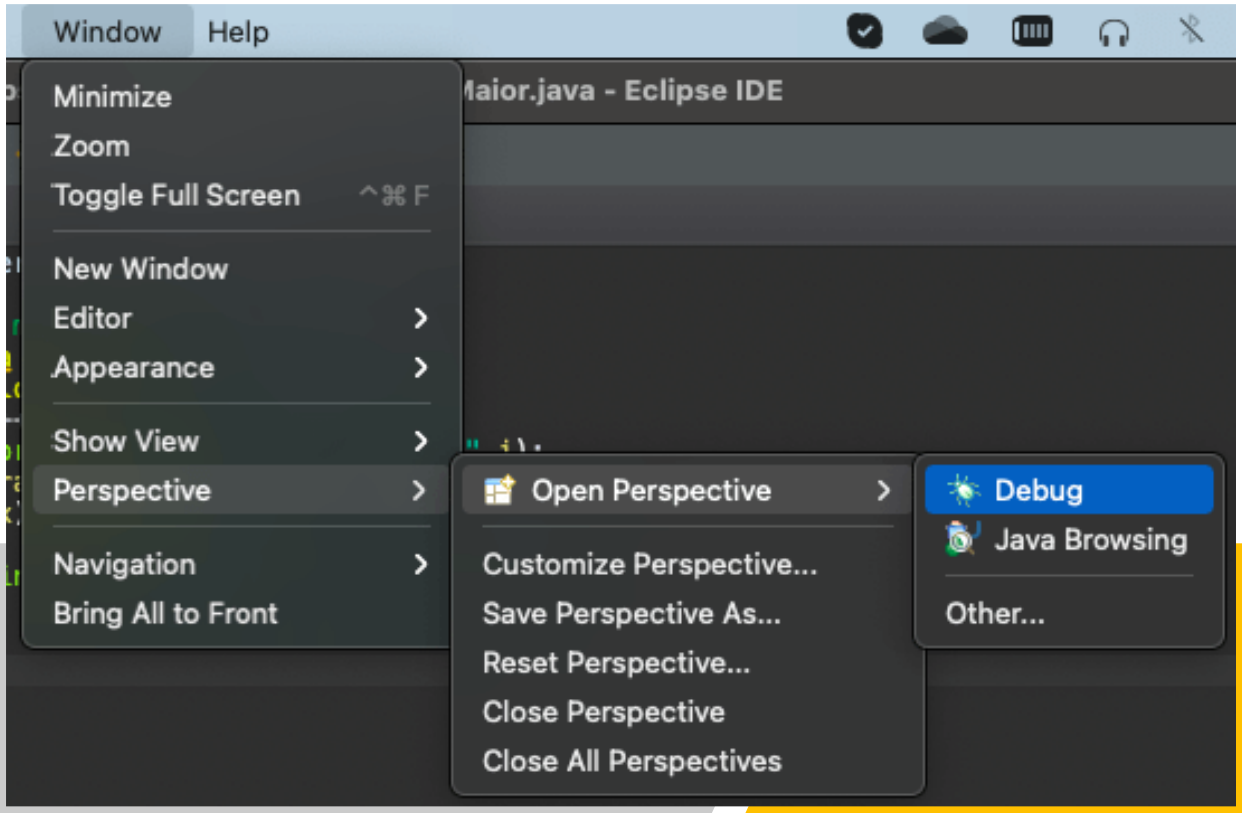
Nesse ponto, vale a pena observarmos um caso em que a depuração é necessáriae, para esse fim, utilizaremos uma aplicação simples em Java e o ambiente integrado de desenvolvimento Eclipse. Originalmente a aplicação deve permitir adigitação de cinco números inteiros e, ao final, informar o maior número digitado. No entanto, o comportamento do programa não é o esperado e alguns recursos de depuração do ambiente de desenvolvimento devem ser usados para que os defeitos sejam encontrados. O Código 3.1 exibe o código-fonte da aplicação.

Anhanguera

```java
import java.util.Scanner;
public class Maior {
        public static void main (String args[]) {
                Scanner entrada = new Scanner(System.in);
            int i, x=0, valor;
            for (i=1;i<7;i++) {
               System.out.printf("\nDigite o %d valor: ",i);
               valor = entrada.nextInt();
               if (valor < x) x = valor;
             }
             System.out.printf("\nO maior valor inserido eh: %d ",x);
        }
}
```

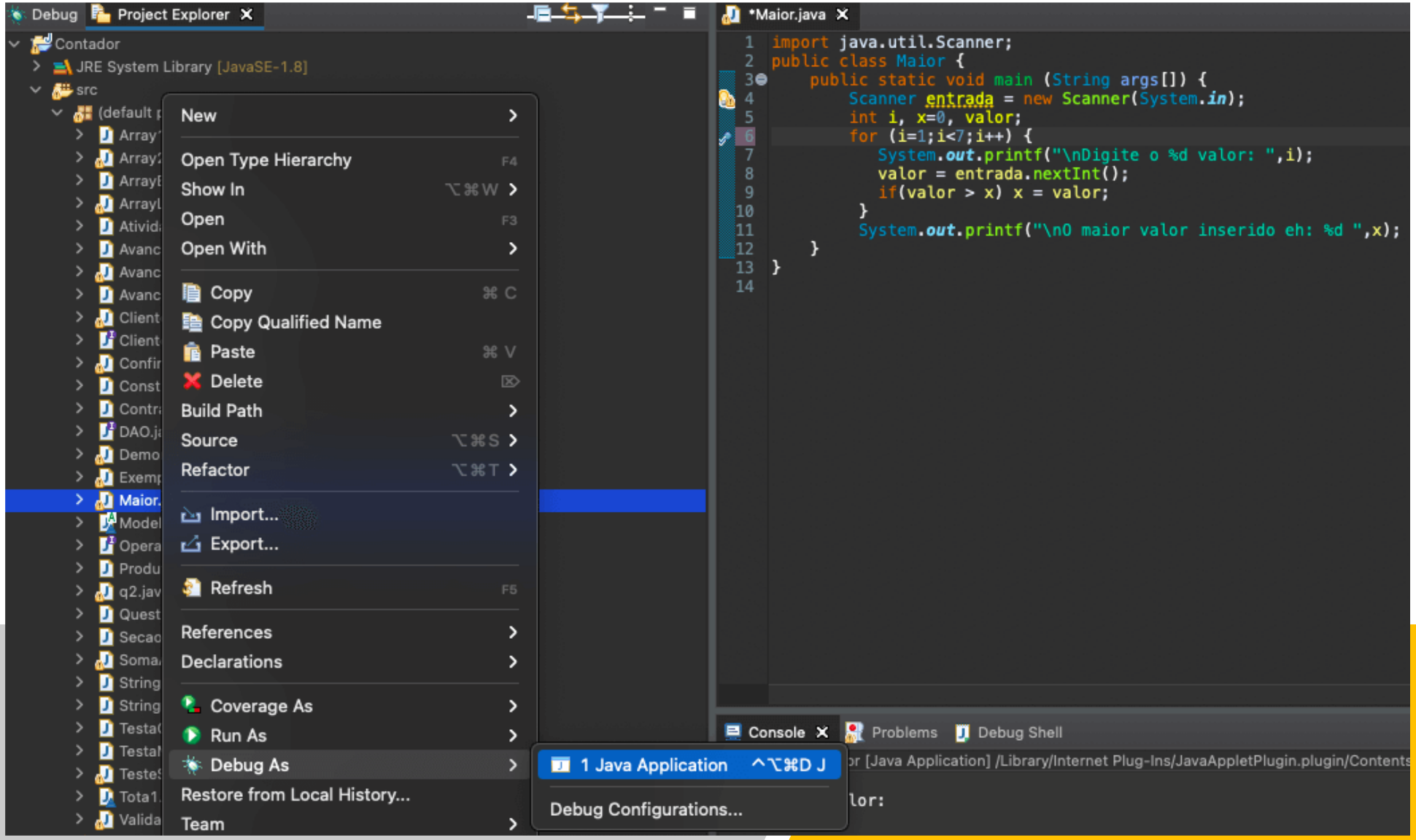
Há dois defeitos nesse código: o primeiro está situado na linha 6: ao invés de permitir a digitação de cinco números, a aplicação solicitará que seis números sejam digitados, dada a comparação feita com valor menor que 7 e não menor que 6. O segundo defeito está situado na linha 9, local em que o teste para apuração domaior valor a cada iteração está invertido. Assim, o valor que retornará a aplicaçãoserá o menor e não o maior, como se espera.

A utilização dos recursos de depuração do Eclipse começa com a mudança daperspectiva de visualização. Na opção de menu Window, acesse Perspective >Open Perspective > Debug, conforme ilustra a Figura 3.5. Essa providência fornece um padrão de dados adequado para a execução da depuração.



O primeiro passo efetivo do processo se dá com a marcação da linha do programa em que se deseja iniciar, de fato, a depuração. Essa ação é realizada por meio doposicionamento do cursor na linha desejada e o acesso à opção Run > Toggle Breakpoint. Uma marca na parte esquerda da linha será criada e, nesse momento, você estará apto a executar a aplicação no modo Debug, conforme mostra a Figura3.6. Posicione o mouse sobre a classe (cujo nome deve aparecer no lado esquerdoda tela), acione o botão direito do mouse sobre ela e acesse a opção Debug As.

Anhanguera


Debug
Project Explorer
Contador
JRE System Library [JavaSE-1.8]
src
(default p
Array1
Array2
ArrayE
ArrayL
Atividi
Avanc
Avanc
Avanc
Client
Client
Confir
Const
Contra
DAO.ja
Demo
Exemp
Maior.
Model
Opera
Produ
q2.jav
Quest
Secac
Soma
String
String
Testa(
Testal
Teste5
Tota1
Valida
New
Open Type Hierarchy F4
Show In ⌥⌘W
Open F3
Open With
Copy ⌘C
Copy Qualified Name
Paste ⌘V
Delete
Build Path
Source ⌥⌘S
Refactor ⌥⌘T
Import...
Export...
Refresh F5
References
Declarations
Coverage As
Run As
Debug As
Restore from Local History...
Team
1 Java Application ^⌥⌘D J
Debug Configurations...
*Maior.java
1 import java.util.Scanner;
2 public class Maior {
3 public static void main (String args[]) {
4 Scanner entrada = new Scanner(System.in);
5 int i, x=0, valor;
6 for (i=1;i<7;i++) {
7 System.out.printf("\nDigite o %d valor: ",i);
8 valor = entrada.nextInt();
9 if(valor > x) x = valor;
10 }
11 System.out.printf("\nO maior valor inserido eh: %d ",x);
12 }
13 }
14
Console
Problems
Debug Shell
[Java Application] /Library/Internet Plug-Ins/JavaAppletPlugin.plugin/Contents
lor:

Neste momento a aplicação será executada no modo de depuração e os valores das variáveis poderão ser verificados passo a passo. Dessa forma, será mais fácilidentificar as anomalias inseridas no código e corrigi-las.

Essa foi a base introdutória de testes que queríamos compartilhar com você. Hámuito ainda a tratar sobre esse procedimento, mas é com a correta compreensãodos conceitos fundamentais que teremos sucesso na continuidade do nosso estudo. Iniciamos a seção tratando da conceituação de verificação e de validação e, na sequência, abordamos o conceito de testes. Depois o situamos como um procedimento e demos a ele uma sequência de etapas. Nesse mesmo contexto,tratamos do plano de testes e demos relevância ao papel do desenvolvedor noprocesso de teste de seu próprio produto. Por fim, abordamos os casos de teste ea depuração como elementos que darão subsídio à continuidade dos nossosestudos sobre teste.

Leia a sentença a seguir:

A _____ é executada ao final de uma etapa do desenvolvimento de um produtoe tem como objetivo determinar se o trabalho que o gerou foi executado _____.Já a _____ corresponde ao processo de avaliação empreendido _____ de oproduto ser disponibilizado ao cliente.

Considerando os conceitos de V&V e os tempos em que são executadas, assinale aalternativa que contenha a sequência correta de termos que completam asentença dada.

a. verificação; corretamente; validação; antes.

b. verificação; rapidamente; validação; depois.

c. validação; rapidamente; verificação; depois.

d. validação; pelas pessoas certas; verificação; depois.

e. verificação; corretamente; verificação; antes.

O procedimento de depuração compõe as ações que visam conferir qualidade aum produto de software. Considerando o conceito, o tempo em que é aplicada eos responsáveis pela aplicação da depuração, analise as afirmações seguintes:

I. Justamente por ter criado o programa, seu desenvolvedor deve permanecerfora do processo de depuração.

II. A depuração é uma das etapas do processo de verificação, já que é feita antesde um certo artefato ser entregue.

III. Embora conte com um procedimento de execução objetivo, a depuração podecarecer de elementos subjetivos de quem a aplica.

É verdadeiro o que se afirma em:

a. I e II apenas.

b. II apenas.

c. I e III apenas.

d. II e III apenas.

e. III apenas.