**Lista de Exercícios 05: Critérios de Testes Estruturais**

1. Considere a classe abaixo:

|  |
| --- |
| public class Calculadora {  /\*\*  \* @param n - inteiro  \* @param valorMaximo - valor maximo que pode ter o somatorio  \* @return - o somatorio de 0 ate |n|, caso somatorio seja <= valorMaximo  \* @throws Exception - caso o somatorio seja > valorMaximo  \*/  public int somatoriaLimitada(int n, int valorMaximo) throws Exception {  int resultado = 0, i = 0;  if(n < 0) {  n = -n;  }    while(i<=n && resultado <= valorMaximo) {  resultado = resultado + i;  i++;  }  if(resultado > valorMaximo)  throw new Exception("valor maximo foi ultrapassado");  else  return resultado;  }  } |

Desenhe o grafo de fluxo de controle (GFC) para o método. Elabore casos de teste em JUnit de forma a alcançar 100% de cobertura do critério de teste estrutural todos-arcos (branch coverage). Comente cada caso de teste com o caminho que foi executado de acordo com o GFC. Enumere os nós no grafo para identificar corretamente os caminhos.

1. Considere a classe abaixo

|  |  |
| --- | --- |
| public class NovaLinha {  /\*\*  \* @param argStr string da qual os caracteres de new line ‘\n’ serão aglutinados  \* @return String  \*/  public String collapseNewlines(String argStr) {  char last = argStr.charAt(0);  StringBuffer sBuf = new StringBuffer();  for (int i = 0; i < argStr.length(); i++) {  char ch = argStr.charAt(i);  if (ch != '\n' || last != '\n') {  sBuf.append(ch);  last = ch;  }  }  return sBuf.toString();  }  } |  |

Considerando o grafo de fluxo de controle (GFC) do método *collapseNewlines,* elabore os casos de teste em JUnit de forma a:

1. Executar os arcos (3,9).
2. A alcançar 100% de cobertura do critério de teste estrutural todos-arcos (branch coverage).
3. Comente cada caso de teste anteriores com o caminho que foi executado de acordo com o GFC.

3) Considere as quatro classes a seguir. Observe que existe um programa original e três mutantes; as mutações aplicadas em cada mutante estão destacadas em negrito.

|  |  |
| --- | --- |
| **Original.java**  public class Original {  //método retorna o maior elemento do  vetor  public int getMaior(int vetor[]) {  int maior = vetor[0];  for (int i = 1; i < vetor.length; i++) {  if(vetor[i] > maior)  maior = vetor[i];  }  return maior;  }  } | **Mutante1.java**  public class Mutante1 {  //método retorna o maior elemento do  vetor  public int getMaior(int vetor[]) {  int maior = vetor[0];  for (int i = 1; i < vetor.length; i++) {  if(vetor[i] **!=** maior)  maior = vetor[i];  }  return maior;  }  } |
| **Mutante2.java**  public class Mutante2 {  //método retorna o maior elemento do  vetor  public int getMaior(int vetor[]) {  int maior = vetor[0];  for (int i = **2**; i < vetor.length; i++) {  if(vetor[i] > maior)  maior = vetor[i];  }  return maior;  }  } | **Mutante3.java**  public class Mutante3 {  //método retorna o maior elemento do  vetor  public int getMaior(int vetor[]) {  int maior = vetor[0];  for (int i = 0; i < vetor.length; i++) {  if(vetor[i] > maior)  maior = vetor[i];  }  return maior;  }  } |

Implemente casos de teste em JUnit para matar cada um dos três mutantes; se algum dos mutantes for equivalente, justifique.

4) Considere as quatro classes a seguir. Observe que existe um programa original e três mutantes; as mutações aplicadas em cada mutante estão destacadas em negrito.

|  |  |
| --- | --- |
| **Original.java**  public class Original {  public int contarA(String word) {  int contador = 0;  for(int i = 0; i < word.length(); i++) {  if(word.charAt(i) == 'a' ||  word.charAt(i) == 'A')  contador++;  }  return contador;  }  } | **Mutante1.java**  public class Mutante1 {  public int contarA(String word) {  int contador = 0;  for(int i = 0; i < word.length(); i++) {  if(word.charAt(i) == 'a' ||  word.charAt(i) == **'a'**)  contador++;  }  return contador;  }  } |
| **Mutante2.java**  public class Mutante2 {  public int contarA(String word) {  int contador = 0;  for(int i = 0; i < word.length(); i++) {  if(word.charAt(i) == 'a' ||  word.charAt(i) == 'A')  contador=**contador+1;**  }  return contador;  }  } | **Mutante3.java**  public class Mutante3 {  public int contarA(String word) {  int contador = 0;  for(int i = 0; i < word.length()**-1**; i++) {  if(word.charAt(i) == 'a' ||  word.charAt(i) == 'A')  contador++;  }  return contador;  }  } |