UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CORNÉLIO PROCÓPIO

Lista de Exercícios: Teste Caixa-Branca
Teste de Mutação

Prof. André Takeshi Endo

(Exercício 7) 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.

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CORNÉLIO PROCÓPIO

(Exercício 11) 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;
   }
}
```

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

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CORNÉLIO PROCÓPIO

(Exercício 14) 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 calcularAnosBissexto(int ano[]) {
        int c = 0;
        for (int i = 0; i < ano.length; i++) {
            if(ano[i] % 400 == 0)
                c++;
            else if(ano[i] % 4 == 0 && ano[i] % 100 != 0)
                c++;
        }
        return c;
    }
}
```

Mutante1.java

```
public class Mutante1 {
    public int calcularAnosBissexto(int ano[]) {
        int c = 0;
        for (int i = 0; i < ano.length; i++) {
            if(ano[i] % 400 == 0)
                c++;
            else if(ano[i] % 4 == 0 && ano[i] % 100 != 0)
                i++;
        }
        return c;
    }
}
```

Mutante2.java

```
public class Mutante2 {
    public int calcularAnosBissexto(int ano[]) {
        int c = 0;
        for (int i = 0; i < ano.length; i++) {
            if(ano[i] % 400 == 0)
                c++;
            else if(ano[i] % 4 == 0 || ano[i] % 100 != 0)
                c++;
        }
        return c;
    }
}
```

Mutante3.java

```
public class Mutante3 {
    public int calcularAnosBissexto(int ano[]) {
        int c = 0;
        for (int i = 0; i < ano.length; i++) {
            if(ano[i] % 40 == 0)
                c++;
            else if(ano[i] % 4 == 0 && ano[i] % 100 != 0)
                c++;
        }
        return c;
    }
}
```

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

(Exercício 16) Considere as cinco classes a seguir. Observe o programa original no lado esquerdo e três mutantes no lado direito. As mutações aplicadas em cada mutante estão destacadas em negrito e identificadas em comentários. Perceba que some as linhas mutadas são apresentadas nos mutantes. O demais trechos de código são iguais ao programa original.

Pessoa.java

```
public class Pessoa {
   private String nome;
   private int idade;

   public Pessoa(String pNome, int pIdade) {
      nome = pNome;
      idade = pIdade;
   }

   public String getNome() { return nome; }

   public int getIdade() { return idade; }
}
```

Original.java

```
public class Original {
   public String definirFaixaEtaria(Pessoa p)
                                 throws RuntimeException {
      if(p.getIdade() < 0 || p.getIdade() >= 110)
         throw new
              IllegalArgumentException("idade invalida");

      int idade = p.getIdade();
      String tipo = "";
      if(idade <= 11)
         tipo = "crianca";
      else if(idade <= 18)
         tipo = "adolescente";
      else if(idade <= 59)
         tipo = "adulto";
      else
         tipo = "idoso";

      return p.getNome() + " eh " + tipo;
   }
}
```

Mutante1.java

```
public class Mutante1 {
   public String definirFaixaEtaria(Pessoa p)
                                 throws RuntimeException {
      if(p.getIdade() < 0 && p.getIdade() >= 110)  // '||' -> '&&'

      //O restante igual ao original
   }
}
```

Mutante2.java

```
public class Mutante2 {
   public String definirFaixaEtaria(Pessoa p)
                                 throws RuntimeException {
      //O restante igual ao original

      else if(idade < 18)     //'<=' -> '<'

      //O restante igual ao original
   }
}
```

Mutante3.java

```
public class Mutante3 {
   public String definirFaixaEtaria(Pessoa p)
                                 throws RuntimeException {
      //O restante igual ao original

      else if(idade == 59)       //'<=' -> '=='

      //O restante igual ao original
   }
}
```

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