Análise Estática - Exercício

Exercício – Análise Estática 2

- **5 Trechos de Programas (alguns reais)**: Identifiquem possíveis anomalias ou violações de padrões de documentação e estruturação do código (e de faltas)
  - identifiquem possíveis regras ou recomendações que não estejam sendo seguidas
- Programas estão redigidos em *Inglês*, mas os padrões de estruturação e documentação são os mesmos
- Programas são implementados em *Java*, mas, em essência, certos padrões são universais, independentes de linguagem
- Código é parcial; somente trecho relevante para o exercício foi mantido

Programa 1 – O que acham da endentação?

```
public static boolean doesMatch( String expression ) {
Stack stack = new Stack( 3 );  boolean match = true; int index = 0; int length =
expression.length();
while( index < length && match ) {
char current = expression.charAt( index++ );
switch( current ) {
case '{' : case '[' : case '(' : stack.push( current );
case '}':
if( stack.pop() != '{' ) match = false; break;
case ']':
if( stack.pop() != '[' )
match = false; break;
case ')':
if( stack.pop() != '(' )
match = false; break;
}
if( ! stack.isEmpty() ) match = false;
}
return match;
}
```

Apresentação adequada?

```
public static boolean compare( String expression ) {
Stack stack = new Stack( 3 );  boolean equality = true; int index = 0; int size = expression.size();
while( index < size && equality ) {
char x = expression.charAt( index++ );
switch( x) {
case '{' : case '[' : case '(' : stack.push( x );
case '}':
if( stack.pop() != '{' ) equality = false; break;
case ']':
if( stack.pop() != '[' )
equality = false; break;
case ')':
if( stack.pop() != '(' )
equality = false; break;
}
if( ! stack.isEmpty() ) equality = false;
}
return equality;
}
```

Este código é escrito de acordo com os padrões de nomes, mas você pode *compreêndo-lo* facilmente?

Você gostaria de produzir *casos de teste* para um método deste tipo?

Você gostaria de ter que *modificar* este código?

Programa 1 – Outros problemas neste código?

```
public static boolean doesMatch( String expression ) {
Stack stack = new Stack( 3 );  boolean match = true; int index = 0; int length =
expression.length();
while( index < length && match ) {
char current = expression.charAt( index++ );
switch( current ) {
case '{' : case '[' : case '(' : stack.push( current );
case '}':
if( stack.pop() != '{' ) match = false; break;
case ']':
if( stack.pop() != '[' )
match = false; break;
case ')':
if( stack.pop() != '(' )
match = false; break;
}
if( ! stack.isEmpty() ) match = false;
}
return match;
}
```

Program 2 – Parte 1...

```
public class SHOP_MANAGER {
    AuthenticationManager authentMan = null;
    AccountManager accountMan = new AccountManager();

    public ShopManager() {
          super();
          authentMan = AuthenticationManager.create();
    }

    public User logon( String user, String password ) throws Exception {
          return authentMan.logon( user, password );
    }

    public void logoff( User user ) throws Exception {
          destroyShoppingCart( user );
          authentMan.logoff( user );
    }

    HashMap shoppingCarts = new HashMap();
    private void destroyShoppingCart( User user ) {
        shoppingCarts.remove( user );
    }

    public void addProductToShoppingCart( User user, Product product, int count )
throws Exception {
          ShoppingCart shoppingCart = getShoppingCart( user );
          shoppingCart.addProduct( product, count );
    }
```

Programa 2 - continuação


```
public class SHOP_MANAGER {
   …
   private ShoppingCart getShoppingCart( User user ) {
         ShoppingCart ret = null;
         if ( shoppingCarts.containsKey( user ) ) {
                  ret = (ShoppingCart)shoppingCarts.get( user );
         } else if ((user.getName()).equals("Admin")) ret==null; user=null;
                  if (!(user.getName()).equals("Admin")) { ret=new ShoppingCart();
                  shoppingCarts.put( user, ret ); }
         }
         return ret;
   }

   public void buyShoppingCart( User user, String creditCardDetails ) throws
   Exception {
         ShoppingCart shoppingCart = getShoppingCart( user );
         double amount = shoppingCart.getTotal();
         accountMan.debit( user, creditCardDetails, amount );
         destroyShoppingCart( user );
   }

   ArrayList productList = new ArrayList();
   public void removeProductFromShop( User user, Product product ) throws Exception
   {
         productList.remove( product );
   }

   public ArrayList getProductList( User user ) throws Exception {
         return productList;
   }
```

Programa 3

- Você percebe algo especial com relação a política de nomes para variáveis?

```
public TupleSpace(String name_,TSCmd tsCmd_,
                  SuperTuple configuration_, SuperTuple accessPermissions_)
                  throws TupleSpaceException {


  if (!(isValidTSName(name_)) )
    throw new TupleSpaceException("BadTSName");
  _tsName       =    name_;
  _tsCmd = tsCmd_;
  Transaction initTrans = null;
  Boolean createOnly = Boolean.FALSE;
  try {
    _tsCmd.command(TSCREATE,
        new Tuple(name_, new Field(SuperTuple.class,configuration_),
                         new Field(SuperTuple.class, accessPermissions_),
                         createOnly),
         _GALAXY_TSNAME, initTrans,0);
  } catch(TupleSpaceException e) {
    Debug.out(0,e.getMessage());
    if (Debug.ON) Debug.out(0,e);
    throw e;
  }
} // TupleSpace
```

Programa 3

- Alguma identificação de boa prática de documentação?

```
public TupleSpace(String name_,TSCmd tsCmd_,
                  SuperTuple configuration_, SuperTuple accessPermissions_)
                  throws TupleSpaceException {
  try {
    _tsCmd.command(TSCREATE,
        new Tuple(name_, new Field(SuperTuple.class,configuration_),
                         new Field(SuperTuple.class, accessPermissions_),
                         createOnly),
        _GALAXY_TSNAME, initTrans,0);
  } catch(TupleSpaceException e) {
    Debug.out(0,e.getMessage());
    if (Debug.ON) Debug.out(0,e);
    throw e;
  } catch(Exception e) {
    Debug.out(0,e);
    throw new TupleSpaceException("TSPACE_EXC0001",e);
  }

} // TupleSpace
```

Programa 4

```
...
switch( grade ) {
   case 'A': case 'a':
      aCount++;
      break;
   case 'B': case 'b':
      bCount++;
      break;
   case 'C': case 'c':
      cCount++;
      break;
   case 'F': case 'f':
      fCount++;
      break;
}
...
```

Programa 5 – Parte 1

```
/**********************************************************************
  ** rollForward.   **
  ********************/
  /**
  *   This method is used to read a checkpoint from disk.
  ***
  *** 02/25/2001 TJL NOTE:  During rewrite of the transaction manager, lock
  *** manager, checkpoint manager and other related components, it has come
  *** to my attention that we were incorrectly using the term "rollback".
  *** That term applies to UNDOing work that was not finished.  In the
  *** context HERE, we need to use the term "ROLL FORWARD", which means that
  …
  *** So, anyway, I'm going to change the name of the method to ROLL FORWARD,
  …
  *** XXXXX Finally, I'd like to thank my Mom for preparing a tasteful cake for
  *** XXXXX me this morning
  ***
  **********************************************************************
  **/
  static TS rollForward( TSServer tsServer_,
 TS galaxy_, String tsName_, String dbType_)
    throws RollbackException
  {
```

Programa 5 – parte 2

```
    // create data base config Tuple
    //   Read the ConfigTuple from the Galaxy Space
    //   Build one if not found
    //  Sorry for this ugly variable name (ct) but I was not felling well today

    ConfigTuple ct = new ConfigTuple();
    if (! tsName_.equals(TupleSpace._GALAXY_TSNAME)&&
! tsName_.equals(TupleSpace._ADMIN_TSNAME)) {

        Tuple config =
          new Tuple("ConfigTuple",tsName_,new Field(ConfigTuple.class));

        SuperTuple found = galaxy_.command( TupleSpace.READ, config,
                                            ignoreClientID,
                                            null, TS._SYSTEM_USER);

        if (found = null) {
          // build a basic Config Tuple.
          ct.setOption(ConfigTuple.TSDBTYPE,dbType_);
          if (dbType_.equals(TupleSpace._DBTYPE_SMALLDB))
            ct.setOption(ConfigTuple.INDEX_FIELD,
                         new Integer(ConfigTuple._DEFAULT_INDEX_FIELD));
        } else {
           // use the previously written ConfigTuple.
           ct = (ConfigTuple)found.getField(2).getValue();
           //

        }
```

Programa 5 – Parte 3

- Algo especial nos comentários do código?

```
// insert every tuple back into the newly created database
SuperTuple tuple = null;
boolean done = false;
while(!done) {
  tuple = (SuperTuple)in.readObject();
  if(!done) {
    int tid = tuple.getTupleID().getUniqueID();
    if ( tid > maxtid)
      maxtid = tid;
      //FIX FIX FIX FIX FIX FIX FIX FIX FIX
      //FIX FIX FIX FIX FIX FIX FIX FIX FIX
      //Toby: review this code line
      tsdb.writeTuple( ignoreClientID, tuple, "rollForward", "Admin");
  }
} // end while
```

Aula 9

Análise Estática - Exercício

Alessandro Garcia
LES/DI/PUC-Rio
Setembro 2010