# Github

January 2, 2019

## Contents

# 1 Account

```
class Account

 instance variables
--  private static usedUsernames: set of Utils`String := {};

  public username: Utils`String;
  public repositories: map Utils`String to Repository := { |-> };

  --public name: Utils`String := [];
  --public description: Utils`String := [];
  --public location: Utils`String := [];
  --public site: Utils`String := [];

 operations

  public Account: Utils`String ==> Account
  Account(un) == (
   username := un;
  );
--  pre un not in set usedUsernames
--  post un in set usedUsernames;


  public newRepository: Utils`String * bool ==> Repository
  newRepository(name, isPriv) == (
   let r = new Repository(name, self, isPriv) in (
    repositories(name) := r;
    return r;
   );
  )
  pre name not in set dom repositories
  post repositories(name).name = name;

end Account
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Account | 15 | 100.0% | 23 |
| newRepository | 22 | 100.0% | 5 |
| Account.vdmpp | | 100.0% | 28 |

# 2 Branch

```
class Branch
 types
 -- TODO Define types here
 values
 -- TODO Define values here
 instance variables
  public name: Utils`String;
  public isProtected: bool;

  private commits: seq of Commit := []; -- Chronologically ordered assured by invariant
```

```
  inv forall i1, i2 in set inds commits &
   i1 < i2 => mk_Date`DateComparable(commits(i1).timestamp) < mk_Date`DateComparable(commits(i2).
       timestamp);

 operations

  public Branch: Utils`String * bool ==> Branch
  Branch(n, prot) == (
   name := n;
   isProtected := prot;
   return self;
  )
  pre n <> []
  post name = n and isProtected = prot;


  public commit: Commit ==> ()
  commit(c) == (
   commits := commits ^ [c];
  )
  pre c not in set elems commits
  post c in set elems commits;


  public getCommits: () ==> seq of Commit
  getCommits() == return commits;

 functions
 -- TODO Define functiones here
 traces
 -- TODO Define Combinatorial Test Traces here
end Branch
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Branch | 16 | 100.0% | 10 |
| commit | 25 | 100.0% | 2 |
| getCommits | 32 | 100.0% | 5 |
| Branch.vdmpp | | 90.0% | 17 |

# 3 Commit

```
class Commit
 types
 -- TODO Define types here
 values
 -- TODO Define values here
 instance variables
  public hash: Utils`String;
  -- public content: Utils`String;
  public timestamp: Date;

  public author: User;

 operations

  public Commit: Utils`String * User * Date ==> Commit
```

```
  Commit(str, u, d) == (
   hash := str;
   author := u;
   timestamp := d;
   return self;
  )
  pre str <> []
  post hash = str;

 functions
 -- TODO Define functions here
 traces
 -- TODO Define Combinatorial Test Traces here
end Commit
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Commit | 14 | 100.0% | 2 |
| Commit.vdmpp | | 100.0% | 2 |

# 4 Date

```
class Date
types
 public DateComparable :: date: Date
 ord mk_DateComparable(d1) < mk_DateComparable(d2) ==
  d1.year < d2.year or
  d1.year = d2.year and d1.month < d2.month or
  d1.year = d2.year and d1.month = d2.month and d1.day < d2.day or
  d1.year = d2.year and d1.month = d2.month and d1.day = d2.day and d1.hour < d2.hour or
  d1.year = d2.year and d1.month = d2.month and d1.day = d2.day and d1.hour = d2.hour and d1.
      minute < d2.minute;

values
 public static startYear = 2000;

instance variables
 private year: nat1;
 private month: nat1;
 private day: nat1;
 private hour: nat;
 private minute: nat;

 inv year >= startYear;
 inv month >= 1 and month <= 12;
 inv day >= 1 and day <= daysOfMonth(year, month);
 inv hour >= 0 and hour < 24;
 inv minute >= 0 and minute < 60;

operations

 public Date: nat1 * nat1 * nat1 * nat * nat ==> Date
 Date(y, mo, d, h, mi) == (
  year := y;
  month := mo;
  day := d;
  hour := h;
  minute := mi;
```

```
  return self;
 ) -- No need for pre conditions -> assured by instance variables invariants
 post year = y and month = mo and day = day and hour = h and minute = mi;

functions

 public static isLeapYear(y: nat1) res: bool == y mod 4 = 0
 pre y >= startYear;


 public static daysOfMonth(y, m: nat1) res : nat == (
  cases m :
   1, 3, 5, 7, 8, 10, 12 -> 31,
   4, 6, 9, 11 -> 30,
   2 -> if isLeapYear(y) then 29 else 28
  end
 )
 pre m >= 1 and m <= 12;

traces
-- TODO Define Combinatorial Test Traces here
end Date
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Date | 28 | 100.0% | 16 |
| daysOfMonth | 43 | 100.0% | 2 |
| isLeapYear | 40 | 100.0% | 4 |
| Date.vdmpp | | 100.0% | 22 |

# 5 Github

```
class Github
types
 -- Map of String (username) to Account
 public AccountsMap = map Utils`String to Account;

values
-- TODO Define values here

instance variables
 public accounts: AccountsMap := { |-> };

operations

 public Github: () ==> Github
 Github() == (return self)
 post card dom accounts = 0;


 public addAccount: Account ==> ()
 addAccount(acc) == (
  accounts(acc.username) := acc;
 )
 pre acc.username not in set dom accounts
 post accounts(acc.username) = acc;
```

```
  public numAccounts: () ==> nat
  numAccounts() == (return card dom accounts);


  public getRepositoriesByTags: set of Tag | set of Utils`String ==> set of Repository
  getRepositoriesByTags(tags) ==
   return {r | r in set dunion {rng a.repositories | a in set rng accounts} & repoMatchesTags(r,
       tags)}
  pre tags <> {};

  -- Get name of the User accounts

  private pure getUsers: () ==> set of Utils`String
  getUsers() == return {un | un in set dom accounts & isofclass(User, accounts(un))}
  post forall un in set RESULT & isofclass(User, accounts(un));


  public pure stargazers: Repository ==> set of Utils`String

  stargazers(repo) ==
   return {un | un in set getUsers() & repo in set narrow_(accounts(un), User).getStars()}
  post (forall un in set RESULT & repo in set narrow_(accounts(un), User).getStars()) and
   (forall un in set getUsers() \ RESULT & repo not in set narrow_(accounts(un), User).getStars()
       );

  -- Gets sequence containing all Repositories
  private getAllRepos: () ==> seq of Repository
  getAllRepos() == (
   dcl reposSet: set of Repository := {}, repos: seq of Repository := [];
   for all acc in set rng accounts do
    reposSet := reposSet union {r | r in set rng acc.repositories \ reposSet};

   for all r in set reposSet do repos := repos ^ [r];
   return repos;
  )
  post forall e in set dunion { rng acc.repositories | acc in set rng accounts } & e in set elems
       RESULT and
   card dunion { rng acc.repositories | acc in set rng accounts } = len RESULT;

  -- Orders Repositories by number of stars using a bubble-sort like algorithm
  public getTopRepos: () ==> seq of Repository
  getTopRepos() == (
   dcl l: seq of Repository := getAllRepos();
   dcl sorted_list: seq of Repository := l;
   for i = len l to 1 by -1 do
    for j = 1 to i-1 do
     if card stargazers(sorted_list(j)) < card stargazers(sorted_list(j+1))
     then (dcl temp: Repository := sorted_list(j);
      sorted_list(j) := sorted_list(j+1);

      sorted_list(j+1) := temp
   );
   return sorted_list;
  )
  post forall i in set {1, ..., len RESULT - 1} & card stargazers(RESULT(i)) >= card stargazers(
      RESULT(i + 1));

 functions
  public static repoMatchesTags(r: Repository, tags: set of Utils`String | set of Tag) res: bool
      ==
  forall t in set tags & (if isofclass(Tag, t) then t.name else t) in set {tInner.name | tInner
      in set r.tags};

end Github
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Github | 13 | 100.0% | 1 |
| addAccount | 17 | 100.0% | 4 |
| getAllRepos | 38 | 0.0% | 0 |
| getRepositoriesByTags | 27 | 100.0% | 6 |
| getTopRepos | 49 | 0.0% | 0 |
| getUsers | 33 | 100.0% | 6 |
| numAccounts | 24 | 100.0% | 3 |
| repoMatchesTags | 65 | 100.0% | 6 |
| stargazers | 37 | 100.0% | 3 |
| Github.vdmpp | | 48.8% | 29 |

# 6 Issue

```
class Issue
types
 -- Map of message ID to Message
 public Messages = map Utils`String to Message;

values
-- TODO Define values here
instance variables
 public id: Utils`String;
 public title: Utils`String;
 public description: Utils`String;

 public messages: Messages := { |-> };

 public assignees: set of User := {};

operations

 public Issue: Utils`String * Utils`String * Utils`String ==> Issue
 Issue(issueID, issueTitle, issueDesc) == (
  id := issueID;
  title := issueTitle;
  description := issueDesc;
  return self;
 )
 post card dom messages = 0;

 --Add a message to this issue

 public addMessage: Message ==> ()
 addMessage(msg) == (
  messages(msg.id) := msg;
 )
 pre not msg.id in set dom messages
 post messages(msg.id) = msg;

 --Assign an user to this issue

 public assignUser: User ==> ()
```

```
 assignUser(user) == (
  assignees := assignees union {user};
  --should user have this issue added to him aswell?
 )
 pre not user in set assignees
 post user in set assignees;


 public numMessages: () ==> nat
 numMessages() == return card dom messages;


 public numAssignees: () ==> nat
 numAssignees() == return card assignees;

functions
-- TODO Define functions here
traces
-- TODO Define Combinatorial Test Traces here
end Issue
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Issue | 18 | 100.0% | 1 |
| addMessage | 28 | 100.0% | 2 |
| assignUser | 36 | 100.0% | 2 |
| numAssignees | 47 | 100.0% | 3 |
| numMessages | 44 | 100.0% | 3 |
| Issue.vdmpp | | 100.0% | 11 |

# 7 Message

```
class Message
types
-- TODO Define types here
values
-- TODO Define values here
instance variables
 public id: Utils`String;
 public content: Utils`String;
 public timestamp: Date;

 public author: User;

operations

 public Message:  Utils`String * Utils`String * User * Date ==> Message
 Message(msgID, cont, auth, date) == (
  id := msgID;
  content := cont;
  author := auth;
  timestamp := date;
  return self;
 );

functions
```

```
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end Message
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Message | 14 | 100.0% | 2 |
| Message.vdmpp | | 100.0% | 2 |

# 8 Organization

```
class Organization is subclass of Account
types
-- TODO Define types here

values
-- TODO Define values here

instance variables
 public members: set of User := {};

operations

 public Organization: Utils`String ==> Organization
 Organization(un) == (
  Account(un);
 );


 public addMember: User ==> ()
 addMember(u) == (
  members := members union {u}
 )
 pre u.username not in set {us.username | us in set members}
 post u in set members;


 public numMembers: () ==> nat
 numMembers() == return card members;

functions
-- TODO Define functions here
traces
-- TODO Define Combinatorial Test Traces here
end Organization
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Organization | 12 | 100.0% | 4 |
| addMember | 17 | 100.0% | 2 |
| numMembers | 24 | 100.0% | 3 |
| Organization.vdmpp | | 100.0% | 9 |

# 9 Release

```
class Release

 instance variables
  public name: Utils`String := [];
  public timestamp : Date;

 operations

  public Release: Utils`String * Date ==> Release
  Release(n, date) == (
   name := n;
   timestamp := date;
   return self;
  )
  pre n <> [];

end Release
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Release | 8 | 100.0% | 2 |
| Release.vdmpp | | 100.0% | 2 |

# 10 Repository

```
class Repository

 instance variables

  public name: Utils`String;
  private isPrivate: bool;
  private description: Utils`String := [];

  private owner: Account;
  private defaultBranch: Branch;
  public tags: set of Tag := {};
  public collaborators: set of User := {};
  public releases: seq of Release := [];
  public branches: map Utils`String to Branch := { |-> };

  inv defaultBranch in set rng branches;
  inv branches(defaultBranch.name) = defaultBranch;

  inv forall i1, i2 in set inds releases &
   i1 < i2 => mk_Date`DateComparable(releases(i1).timestamp) < mk_Date`DateComparable(releases(i2
       ).timestamp);

 operations

  public Repository: Utils`String * Account * bool ==> Repository
  Repository(n, acc, priv) == (
   name := n;
   isPrivate := priv;
   owner := acc;
```

```
 if isofclass(User, owner) then collaborators := {owner};

 let master = new Branch("master", true) in ( -- Github creates default branch master
  defaultBranch := master;
  branches := { "master" |-> master };
 );
 return self;
)
pre n <> []
post name = n and isPrivate = priv and owner = acc and
 (isofclass(Organization, owner) or (isofclass(User, owner) and owner in set collaborators))
     and
 defaultBranch.name = "master" and card dom branches = 1;



public addRelease: Release ==> ()
addRelease(rel) == (
 releases := releases ^ [rel];
)
pre rel.name not in set {r.name | r in seq releases} -- and rel.timestamp > releases[-1].
    timestamp
post releases(len releases) = rel;


public addTag: Tag ==> ()
addTag(tag) == tags := tags union {tag}
post tag in set tags;


public createBranch: Utils`String * bool ==> Branch
createBranch(n, prot) == (
 let b = new Branch(n, prot) in (
  branches(n) := b;
  return b;
 );
)
pre n not in set dom branches
post let b = branches(n) in b.name = n and b.isProtected = prot;


public commit: User * Utils`String * Utils`String * Date ==> ()
commit(usr, branchName, hash, date) == (
 branches(branchName).commit(new Commit(hash, usr, date));
)
pre (usr in set collaborators or not isPrivate) and
 branchName in set dom branches;


public addCollaborator: User ==> ()
addCollaborator(usr) == collaborators := collaborators union {usr}
post usr in set collaborators;

-- Getters

public getDefaultBranch: () ==> Branch
getDefaultBranch() == return defaultBranch;


public getDescription: () ==> Utils`String
getDescription() == return description;


public numReleases: () ==> nat
numReleases() == return len releases;
```

```
  public isRepoPrivate: () ==> bool
  isRepoPrivate() == return isPrivate;

  -- Setters

  public setDefaultBranch: Utils`String ==> ()
  setDefaultBranch(bName) == defaultBranch := branches(bName)
  pre bName in set dom branches
  post defaultBranch.name = bName;


  public setDescription: Account * Utils`String ==> ()
  setDescription(acc, desc) == description := desc
  pre acc = owner
  post description = desc;


  public setPrivacy: Account * bool ==> ()
  setPrivacy(acc, privacy) == isPrivate := privacy
  pre acc = owner
  post isPrivate = privacy;

end Repository
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Repository | 23 | 100.0% | 9 |
| addCollaborator | 70 | 100.0% | 3 |
| addRelease | 42 | 100.0% | 2 |
| addTag | 49 | 100.0% | 5 |
| commit | 63 | 100.0% | 2 |
| createBranch | 53 | 100.0% | 1 |
| getDefaultBranch | 75 | 100.0% | 11 |
| getDescription | 78 | 100.0% | 2 |
| isRepoPrivate | 84 | 100.0% | 3 |
| numReleases | 81 | 100.0% | 3 |
| setDefaultBranch | 88 | 100.0% | 2 |
| setDescription | 93 | 100.0% | 2 |
| setPrivacy | 98 | 100.0% | 1 |
| Repository.vdmpp | | 100.0% | 46 |

# 11 Tag

```
class Tag

 instance variables
  public name: Utils`String;
  --public description: Utils`String := [];

 operations

  public Tag: Utils`String ==> Tag
```

```
  Tag(tag) == (
   name := tag;
   return self;
  );

end Tag
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Tag | 8 | 100.0% | 4 |
| Tag.vdmpp | | 100.0% | 4 |

# 12 User

```
class User is subclass of Account
types
-- TODO Define types here

values
-- TODO Define values here

instance variables
 private selfUser: User; -- For invariant purposes
 private followers: set of User := {};
 private following: set of User := {};
 private stars: set of Repository := {};

 inv selfUser not in set followers; -- makes pre self <> user unecessary
 inv selfUser not in set following; -- makes pre self <> user unecessary

operations

 public User: Utils`String ==> User
 User(un) == (
--  username := un;
  selfUser := self;
  Account(un);
 );


 private addFollower: User ==> ()
 addFollower(follower) == (
  followers := followers union {follower};
 )
 post follower in set followers;


 private removeFollower: User ==> ()
 removeFollower(us) == (
  followers := followers \ {us};
 )
 post us not in set followers;


 public follow: User ==> ()
 follow(us) == (
  following := following union {us};
  us.addFollower(self);
```

```
 )
 post us in set self.following and
    self in set us.followers;


 public unfollow: User ==> ()
 unfollow(us) == (
  following := following \ {us};
  us.removeFollower(self);
 )
 pre us in set following and self in set us.followers
 post us not in set following and self not in set us.followers;


 public clearFollowing: () ==> ()
 clearFollowing() == (
  for all us in set following
   do unfollow(us)
 )
 post card following = 0;


 public star: Repository ==> ()
 star(repo) == stars := stars union {repo}
 pre repo not in set stars
 post repo in set stars;


 public unstar: Repository ==> ()
 unstar(repo) == stars:= stars \ {repo}
 pre repo in set stars
 post repo not in set stars;

 -- Getters

 public getFollowing: () ==> set of User
 getFollowing() == return following;


 public getFollowers: () ==> set of User
 getFollowers() == return followers;


 public pure getStars: () ==> set of Repository
 getStars() == return stars;

functions
-- TODO Define functions here
traces
-- TODO Define Combinatorial Test Traces here
end User
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| User | 18 | 100.0% | 19 |
| addFollower | 25 | 100.0% | 1 |
| clearFollowing | 53 | 100.0% | 1 |
| follow | 37 | 100.0% | 1 |
| getFollowers | 74 | 100.0% | 2 |
| getFollowing | 71 | 100.0% | 2 |

| | | | |
|---|---|---|---|
| getStars | 77 | 100.0% | 21 |
| removeFollower | 31 | 100.0% | 1 |
| star | 60 | 100.0% | 3 |
| unfollow | 45 | 100.0% | 1 |
| unstar | 65 | 100.0% | 2 |
| User.vdmpp | | 100.0% | 54 |

# 13 Utils

```
class Utils
 types
  public String = seq of char;

 values
 -- TODO Define values here
 instance variables
 operations

  bubbleSort: seq of nat ==> seq of nat
  bubbleSort(l) == (
   dcl sorted_list : seq of nat := l;
   for i = len l to 1 by -1 do
    for j = 1 to i-1 do
     if sorted_list(j) > sorted_list(j+1)
      then (dcl temp:nat := sorted_list(j);
       sorted_list(j) := sorted_list(j+1);
       sorted_list(j+1) := temp
   );
   return sorted_list;
  );

 functions

  public static min(s: set of nat) res: nat ==
  iota n1 in set s & forall n2 in set s & n1 <= n2
  pre card s > 0;


  public static max(s: set of nat) res: nat ==
  iota n1 in set s & forall n2 in set s & n1 >= n2
  pre card s > 0;


  public static isAscendingOrder[@T](s: seq of @T) res: bool ==
  forall i in set {1, ..., len s - 1} & s(i) <= s(i+1);

 traces
 -- TODO Define Combinatorial Test Traces here
end Utils
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| bubbleSort | 9 | 0.0% | 0 |
| isAscendingOrder | 31 | 0.0% | 0 |
| max | 27 | 0.0% | 0 |

| | | | |
|---|---|---|---|
| min | 23 | 0.0% | 0 |
| Utils.vdmpp | | 0.0% | 0 |

# 14 DateTest

```
class DateTest
types
-- TODO Define types here
values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
operations

  private testIsLeapYear: () ==> ()
  testIsLeapYear() == (
   TestUtils`assertTrue(Date`isLeapYear(2020));
   TestUtils`assertFalse(Date`isLeapYear(2018));
  );


  private testDaysOfMonth: () ==> ()
  testDaysOfMonth() == (
   TestUtils`assertTrue(Date`daysOfMonth(2020, 2) = 29);
   TestUtils`assertTrue(Date`daysOfMonth(2018, 2) = 28);
   TestUtils`assertTrue(Date`daysOfMonth(2018, 12) = 31);
   TestUtils`assertTrue(Date`daysOfMonth(2018, 4) = 30);
  );


  private testCompareDates: () ==> ()
  testCompareDates() == (
   TestUtils`assertTrue(
    mk_Date`DateComparable(new Date(2010, 3, 12, 20, 11)) <
    mk_Date`DateComparable(new Date(2012, 4, 21, 23, 56))
   );
   TestUtils`assertTrue(
    mk_Date`DateComparable(new Date(2010, 3, 12, 20, 11)) <
    mk_Date`DateComparable(new Date(2010, 4, 21, 23, 56))
   );
   TestUtils`assertTrue(
    mk_Date`DateComparable(new Date(2010, 3, 12, 20, 11)) <
    mk_Date`DateComparable(new Date(2010, 3, 21, 23, 56))
   );
   TestUtils`assertTrue(
    mk_Date`DateComparable(new Date(2010, 3, 12, 20, 11)) <
    mk_Date`DateComparable(new Date(2010, 3, 12, 23, 56))
   );
   TestUtils`assertTrue(
    mk_Date`DateComparable(new Date(2010, 3, 12, 20, 11)) <
    mk_Date`DateComparable(new Date(2010, 3, 12, 20, 56))
   );
  );


  public static main: () ==> ()
  main() == (
   let dt = new DateTest() in (
    dt.testIsLeapYear();
    dt.testDaysOfMonth();
```

```
    dt.testCompareDates();
   );
  );


functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end DateTest
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| main | 47 | 100.0% | 1 |
| testCompareDates | 23 | 100.0% | 1 |
| testDaysOfMonth | 15 | 100.0% | 1 |
| testIsLeapYear | 9 | 100.0% | 1 |
| DateTest.vdmpp | | 100.0% | 4 |

# 15 GithubTest

```
class GithubTest

 values
  firstUsername = "sample-username";
  secondUsername = "different-username";
  thirdUsername = "other-username";
  fourthUsername = "another-one";

  tagFEUP = new Tag("FEUP");
  tagVDM = new Tag("VDM");

 instance variables
  gh : Github := new Github();

 operations

  private testAddAccount: () ==> ()
  testAddAccount() ==
  (
   TestUtils`assertTrue(gh.numAccounts() = 0);
   gh.addAccount(new User(firstUsername));
   TestUtils`assertTrue(gh.numAccounts() = 1);
   gh.addAccount(new Organization(secondUsername));
   TestUtils`assertTrue(gh.numAccounts() = 2);
  );


  private testGetReposByTags: () ==> ()
  testGetReposByTags() ==
  (
   let r1 = gh.accounts(firstUsername).newRepository("MFES", true),
     r2 = gh.accounts(secondUsername).newRepository("OVERTURE", true) in (
    r1.addTag(tagFEUP);
    r1.addTag(tagVDM);
    r2.addTag(tagVDM);
```

```
    TestUtils`assertTrue(r2 not in set gh.getRepositoriesByTags({tagFEUP}));
    TestUtils`assertTrue(r2 in set gh.getRepositoriesByTags({tagVDM}));
    TestUtils`assertTrue(r2 in set gh.getRepositoriesByTags({"VDM"}));
   );
  );


  private testStargazers: () ==> ()
  testStargazers() ==
  (
   let u3 = new User(thirdUsername), u4 = new User(fourthUsername), r = u3.newRepository("VDM++",
        true) in (
    gh.addAccount(u3); gh.addAccount(u4);
    u3.star(r);
    TestUtils`assertTrue(gh.stargazers(r) = {thirdUsername});
    u4.star(r);
    TestUtils`assertTrue(gh.stargazers(r) = {thirdUsername, fourthUsername});
    u3.unstar(r);
    TestUtils`assertTrue(gh.stargazers(r) = {fourthUsername});
   );
  );


  private testGetTopRepos: () ==> ()
  testGetTopRepos() == (
   TestUtils`assertTrue(len gh.getTopRepos() = 3);
   TestUtils`assertTrue(gh.getTopRepos()(1).name = "VDM++");
  );


  public static main: () ==> ()
  main() ==
  (
   let gt = new GithubTest() in (
    gt.testAddAccount();
    gt.testGetReposByTags();
    gt.testStargazers();
--    gt.testGetTopRepos(); -- TODO call when getTopRepos is fixed
   );
  );


end GithubTest
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| main | 61 | 100.0% | 1 |
| testAddAccount | 16 | 100.0% | 1 |
| testGetReposByTags | 26 | 100.0% | 1 |
| testGetTopRepos | 55 | 0.0% | 0 |
| testStargazers | 41 | 100.0% | 1 |
| GithubTest.vdmpp | | 90.3% | 4 |

# 16 IssueTest

```
class IssueTest
```

```
types
-- TODO Define types here
values
-- TODO Define values here
instance variables
 issue: Issue := new Issue("#7400", "test title", "test description")
operations

 private testAddMessage: () ==> ()
 testAddMessage() == (
  TestUtils`assertTrue(issue.numMessages() = 0);
  issue.addMessage(new Message("msgID", "Content", new User("username"), new Date(2018, 12, 30,
      22, 28)));
  TestUtils`assertTrue(issue.numMessages() = 1);
  issue.addMessage(new Message("msgID2", "Content2", new User("username2"), new Date(2018, 12,
      30, 22, 29)));
  TestUtils`assertTrue(issue.numMessages() = 2);
 );


 private testAsignUser: () ==> ()
 testAsignUser() == (
  TestUtils`assertTrue(issue.numAssignees() = 0);
  issue.assignUser(new User("username"));
  TestUtils`assertTrue(issue.numAssignees() = 1);
  issue.assignUser(new User("username2"));
  TestUtils`assertTrue(issue.numAssignees() = 2);
 );


 public static main: () ==> ()
 main() == (
  let i = new IssueTest() in (
   i.testAddMessage();
   i.testAsignUser();
  );
 );

functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end IssueTest
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| main | 27 | 100.0% | 1 |
| testAddMessage | 9 | 100.0% | 1 |
| testAsignUser | 18 | 100.0% | 3 |
| IssueTest.vdmpp | | 100.0% | 5 |

# 17 OrganizationTest

```
class OrganizationTest
 types
 -- TODO Define types here
 values
```

```
 -- TODO Define values here
 instance variables
  org : Organization := new Organization("FEUP");
 operations

  private testAddMember: () ==> ()
  testAddMember() == (
   TestUtils`assertTrue(org.numMembers() = 0);
   org.addMember(new User("sample-username"));
   TestUtils`assertTrue(org.numMembers() = 1);
   org.addMember(new User("different-username"));
   TestUtils`assertTrue(org.numMembers() = 2);
  );


  public static main: () ==> ()
  main() ==
  (
   new OrganizationTest().testAddMember();
  );
 functions
 -- TODO Define functions here
 traces
 -- TODO Define Combinatorial Test Traces here
end OrganizationTest
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| main | 18 | 100.0% | 2 |
| testAddMember | 9 | 100.0% | 1 |
| OrganizationTest.vdmpp | | 100.0% | 3 |

# 18 RepositoryTest

```
class RepositoryTest
types
-- TODO Define types here
values
-- TODO Define values here
instance variables
 o : Account := new Organization("feup");
 r : Repository := new Repository("mfes", o, true);

operations

 private testConstructor: () ==> ()
 testConstructor() == (
  let org = new Organization("org"), repo = new Repository("test", org, true) in (
   TestUtils`assertTrue(repo.name = "test");
   TestUtils`assertTrue(repo.isRepoPrivate());
   TestUtils`assertTrue(card repo.collaborators = 0);
  );

  let usr = new User("usr"), repo = new Repository("test2", usr, false) in (
   TestUtils`assertTrue(card repo.collaborators = 1);
  );
```

```
 TestUtils`assertTrue(r.getDefaultBranch().name = "master");
 TestUtils`assertTrue(card dom r.branches = 1);
 TestUtils`assertTrue(r.branches("master") = r.getDefaultBranch());
);


private testSetDescription: () ==> ()
testSetDescription() == (
 r.setDescription(o, "Projeto de MFES");
 TestUtils`assertTrue(r.getDescription() = "Projeto de MFES");

 r.setDescription(o, "description");
 TestUtils`assertTrue(r.getDescription() = "description");
);


private testAddCollaborator: () ==> ()
testAddCollaborator() == (
 let u1 = new User("one"), u2 = new User("two") in (
  r.addCollaborator(u1);
  TestUtils`assertTrue(r.collaborators = {u1});

  r.addCollaborator(u2);
  TestUtils`assertTrue(r.collaborators = {u1, u2});
 );
);


private testAddTag: () ==> ()
testAddTag() == (
 let tag1 = new Tag("AI"), tag2 = new Tag("WebDev") in (
  r.addTag(tag1);
  TestUtils`assertTrue(r.tags = {tag1});

  r.addTag(tag2);
  TestUtils`assertTrue(r.tags = {tag1, tag2});
 );
);


private testCreateBranch: () ==> ()
testCreateBranch() == (
 let branch = r.createBranch("develop", true), b = r.branches("develop") in (
  TestUtils`assertTrue(branch.name = b.name);
  TestUtils`assertTrue(branch.name = "develop");

  TestUtils`assertTrue(branch.isProtected = b.isProtected);
  TestUtils`assertTrue(branch.isProtected);
 );
 TestUtils`assertTrue(card dom r.branches = 2);
);


private testSetDefaultBranch: () ==> ()
testSetDefaultBranch() == (
 TestUtils`assertTrue(r.getDefaultBranch().name = "master");
 r.setDefaultBranch("develop");
 TestUtils`assertTrue(r.getDefaultBranch().name = "develop");
);


private testCommit: () ==> ()
testCommit() == (
 let usr = new User("contributor"), pub = new Repository("public", o, false) in (
  r.addCollaborator(usr);
```

```
   TestUtils`assertTrue(len r.getDefaultBranch().getCommits() = 0);
   r.commit(usr, r.getDefaultBranch().name, "hash", new Date(2018, 12, 30, 22, 19));
   TestUtils`assertTrue(len r.getDefaultBranch().getCommits() = 1);

   -- Can also contribute to public repositories
   pub.commit(usr, pub.getDefaultBranch().name, "hash", new Date(2018, 12, 30, 22, 20));
   TestUtils`assertTrue(len pub.getDefaultBranch().getCommits() = 1);
  );
 );


 private testCommitHistory: () ==> ()
 testCommitHistory() == (
  TestUtils`assertTrue(len r.getDefaultBranch().getCommits() = 1);
  r.setDefaultBranch("master");
  TestUtils`assertTrue(len r.getDefaultBranch().getCommits() = 0);
 );


 private testSetPrivacy: () ==> ()
 testSetPrivacy() == (
  TestUtils`assertTrue(r.isRepoPrivate());
  r.setPrivacy(o, false);
  TestUtils`assertFalse(r.isRepoPrivate());
 );


 private testAddRelease: () ==> ()
  testAddRelease() ==
  (
   TestUtils`assertTrue(r.numReleases() = 0);
   r.addRelease(new Release("v1.1", new Date(2018, 12, 30, 22, 28)));
   TestUtils`assertTrue(r.numReleases() = 1);
   r.addRelease(new Release("v1.2", new Date(2018, 12, 30, 22, 29)));
   TestUtils`assertTrue(r.numReleases() = 2);
  );


 public static main: () ==> ()
 main() == (
  let rt = new RepositoryTest() in (
   rt.testConstructor();
   rt.testSetDescription();
   rt.testAddCollaborator();
   rt.testAddTag();
   rt.testAddRelease();
   rt.testCreateBranch();
   rt.testSetDefaultBranch();
   rt.testCommit();
   rt.testCommitHistory();
   rt.testSetPrivacy();
  );
 );

functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end RepositoryTest
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|

| | | | |
|---|---|---|---|
| main | 116 | 100.0% | 1 |
| testAddCollaborator | 37 | 100.0% | 1 |
| testAddRelease | 106 | 100.0% | 1 |
| testAddTag | 48 | 100.0% | 1 |
| testCommit | 78 | 100.0% | 1 |
| testCommitHistory | 92 | 100.0% | 1 |
| testConstructor | 11 | 100.0% | 1 |
| testCreateBranch | 59 | 100.0% | 1 |
| testSetDefaultBranch | 71 | 100.0% | 1 |
| testSetDescription | 28 | 100.0% | 1 |
| testSetPrivacy | 99 | 100.0% | 1 |
| RepositoryTest.vdmpp | | 100.0% | 11 |

# 19 TestAll

```
class TestAll

 operations

  public static main: () ==> ()
  main() ==
  (
   GithubTest`main();
   OrganizationTest`main();
   RepositoryTest`main();
   UserTest`main();
   IssueTest`main();
   DateTest`main();
  );

end TestAll
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| main | 4 | 100.0% | 1 |
| TestAll.vdmpp | | 100.0% | 1 |

# 20 TestUtils

```
class TestUtils
 types
 -- TODO Define types here
 values
 -- TODO Define values here
 instance variables
 -- TODO Define instance variables here

 operations

  public static assertTrue: bool ==> ()
```

```
   assertTrue(cond) == return
   pre cond;


   public static assertFalse: bool ==> ()
   assertFalse(cond) == return
   pre not cond;


   public static randomUser: seq of User ==> User
   randomUser(users) == (
    return users(MATH`rand(len users) + 1)
   );

 functions
--  public static randomElement[@T](s: set of @T) res: @T ==
--  [e | e in set s](MATH`rand(card s));

 traces
 -- TODO Define Combinatorial Test Traces here
end TestUtils
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| assertFalse | 14 | 100.0% | 3 |
| assertTrue | 10 | 100.0% | 124 |
| randomUser | 18 | 100.0% | 2 |
| TestUtils.vdmpp | | 100.0% | 129 |

# 21 UserTest

```
class UserTest

 instance variables
  private users : seq of User := [new User("n") | dummy in set {1, ..., 5}];
  private user : User := new User("username");

 operations

  private testFollowUnfollow: () ==> ()
  testFollowUnfollow() == (
   let u = TestUtils`randomUser(users) in (
    user.follow(u);
    TestUtils`assertTrue(u in set user.getFollowing() and user in set u.getFollowers());
    user.clearFollowing();
    TestUtils`assertFalse(u in set user.getFollowing() or user in set u.getFollowers());
   )
  );


  private testNewRepo: () ==> ()
  testNewRepo() == (
   let r = user.newRepository("FEUP-MFES", true) in (
    TestUtils`assertTrue(user.repositories(r.name) = r)
   );
  );
```

```
  private testStarUnstar: () ==> ()
  testStarUnstar() == (
   let r = (TestUtils`randomUser(users)).newRepository("FEUP-MFES", true) in (
    TestUtils`assertTrue(user.getStars() = {});
    user.star(r);
    TestUtils`assertTrue(user.getStars() = {r});
    user.unstar(r);
    TestUtils`assertTrue(user.getStars() = {});
   );
  );


  public static main: () ==> ()
  main() == (
   let ut = new UserTest() in (
    ut.testFollowUnfollow();
    ut.testNewRepo();
    ut.testStarUnstar();
   );
  );

 traces
  FollowUnfollow :
   user.clearFollowing();
   (let u = TestUtils`randomUser(users) in user.follow(u)){1, 5};
   (let u = TestUtils`randomUser(users) in user.unfollow(u)){1, 5};

end UserTest
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| main | 36 | 100.0% | 1 |
| testFollowUnfollow | 8 | 100.0% | 1 |
| testNewRepo | 18 | 100.0% | 1 |
| testStarUnstar | 25 | 100.0% | 1 |
| UserTest.vdmpp | | 90.7% | 4 |