نکاتی برای نوشتن مستندات[1] خوب

مستندات در واقع جای‌گزین تحویل حضوری هستند. در نتیجه باید طوری باشند که اولاً راحت قابل فهم باشند، ثانیاً نکات کلیدی و نابدیهی کار شما را شامل باشند. فرض کنید می‌خواهید پروژه‌ای که انجام داده‌اید را به طور خلاصه برای یک نفر ناوارد توضیح دهید. چیزهایی که به او می‌گوئید را باید در مستندات بنویسید و چیزهایی که خودش بدون فکر زیاد می‌تواند بفهمد را لازم نیست.
و ضمناً هم برای درک بهتر موضوع و هم بیان دقیق جزئیات، مثال زدن خیلی می‌تواند مفید باشد، مخصوصاً زدن مثال‌های خاص.
مثلاً در فاز سوم پروژه‌ی کامپایلر، در طراحی جدول نمادها[2] نکات نابدیهی‌ای که وجود داشت این‌ها بود: ساختار کلی جدول، یعنی از چه کلاس‌هایی استفاده کرده‌اید و نقش آن‌ها چیست؟ جزئیات نابدیهی این‌ها هستند: چگونگی پیاده‌سازی extends و چگونگی نگه‌داری آرگومان‌های توابع.

مثال خوب:

۱. MethodTable شامل اجزاي زير است :

- يك HashTable<String , String> به نام allVars است كه به ازاي هر متغير تعريف شده در كلاس ، نام آن به همراه type آن را به allVars اضافه مي‌كنيم) .مولفه ی اول مربوط به نام و مولفه ی دوم مربوط به نوع متغير است).

-يک HashTable<String , String> به نام allParams است كه به ازاي هر parameter تعريف شده در كلاس، نام آن به همراه type آن را به allParams اضافه مي‌كنيم.

- و همين طور يک ArrayList<String> به نام params كه ترتيب پارامترها را در خود نگه مي‌دارد .وقتی متدی صدا زده می شود، به کمک این لیست درست بودن ترتیب پارامترهای داده شده به آن را چک می‌کنیم.

- Parent كه نام كلاسي كه method در آن تعريف شده را در خود نگه مي‌دارد.

- returnType كه براي هر تابع مشخص شده را نگاه می‌دارد.

مثال بد:

۱. در این فاز برای ساختن جدول سمبل‌ها از کلاس SymbolTableBuilder استفاده می‌شود. این کلاس گره‌های درخت را پرمایش کرده و اسم و تایپ سمبل‌ها را استخراج می‌کند. این

---

[1] documentation
[2] symbol table

کلاس از سه کلاس اصلی classSymbol و methodSymbol و fieldSymbol برای شناسایی سمبل "کلاس"،"متد" و "فیلد"های کلاس‌های miniJava کمک می‌گیرد. در همین مرحله اگر اسم دو کلاس در برنامه یکی باشد و یا اینکه دو متد هم‌نام یا فیلد هم‌نام داشته‌باشیم، semanticError داده‌می‌شود و بقیه‌ی مراحل انجام نمی‌شود. ← فاقد جزئیات کافی

۲. هر عضو سیمبل تیبل ۴ عضو دارد ، اولی نشان‌دهنده ی کلاسی است که در آن هستیم ، دومی نشان دهنده ی تابعی که در آن هستیم ، سومی نام این سیمبل است و آخری تایپ آن. (البته در مواقعی مانند وقتی که سیمبل یک کلاس است ، تفاوت‌های اندکی وجود دارد) ← فاقد جزئیات کافی

درباره‌ی جزئیات انجام type-checking:

مثال بد:

۱. طریقه‌ی تایپ چکینگ ، کاملن معمولی است و به صورتِ بازگشتی روی درختِ برنامه انجام می شود. مثلا برای یک عبارت جمع ، ابتدا چک می شود که تایپ سمت چپ اینتیجر باشد ، سپس چک می شود که تایپ سمت راست اینتیجر باشد و در انت‌ها هم تایپ اینتیجر به عنوانِ تایپ کلی عبارت برگردانده می شود. ← فاقد جزئیات کافی

۲. تابع visit را صدا می زنیم که این تابع کار type checking را انجام می دهد که در این تابع duplication را نیز check می کنیم و هم چنین اینکه آیا متغیر تعریف شده یا نه ، یا اینکه متغیر جز فیلد های کلاس هست یا نه . هم چنین برای check ، method call ها ، field args را که در traverse ساخته بودیم با type ، پارامترهای ورودی چک می کنیم ← فاقد جزئیات کافی

۳. در زمان Parse، Type متغیرها به صورت L-Attribute در گره‌های درخت منتشر می‌شوند تا نهایتاً در Pass مربوطه به Type Check بر مبنای Attribute های تولید شده کنترل‌های لازم صورت گیرند. ← فاقد جزئیات کافی

مثال خوب:

۱. برای عمل Type-Checking لازم است یک بار درخت را طی کنیم. برای این کار برای هر node درخت به جز Expressionها یک متد با نام typeCheck تعریف شده است که در صورت وجود مشکل یک Exception از نوع TypeException را Throw می‌کند. برای

Expressionها نیز یک متد با نام getType تعریف شده است که Type آن Expression را برمی‌گرداند و در صورت وجود مشکل یک Exception از نوع TypeException را Throw می‌کند. علت اینکه برای Expressionها متد متفاوتی تهیه شده است آن است که Expressionها دارای Type هستند اما سایر nodeهای درخت Type ندارند. در طی کردن درخت از attribute استفاده نشده است.

۲. برای این کار از کلاس TypeChecker که با توجه به الگوی Visitor پیاده‌سازی شده، استفاده می‌شود. تابع visit(node, arg)، برای node های عبارت (ExpressionNode، IdentifierNode و Node های مربوط به جمع، ضرب و...) نوع آن‌ها را بازمی‌گرداند. بدین وسیله عملیات Type Checking به‌سادگی قابل انجام است. مثلا در فراخوانی visit() برای یک Node مقایسه، ابتدا دو طرف این عملگر visit() می‌شوند. نوع آن‌ها باید int باشد. سپس این تابع نوع boolean را به خروجی می‌دهد.