X

УчИМИ Ученически институт по математика и информатика

ДЕВЕТНАДЕСЕТА УЧЕНИЧЕСКА КОНФЕРЕНЦИЯ

УК’19

ТЕМА НА ПРОЕКТА

РАЗРАБОТКА НА ПРОГРАМЕН ЕЗИК

Автор(и):

Цветелин Костадинов Цецков,

ученик в ПМПГ „Св. Климент Охридски“- Монтана, 11 б клас

Дейвид Йорданов Каменов,

ученик в ПМПГ „Св. Климент Охридски“- Монтана, 11 б клас

(трите имена, училище, град, клас)

Научен ръководител (консултант):

Илиян Ценков,

учител по Информатика и Информационни технологии,
ПМПГ “Св. Климент Охридски“- Монтана

(име, фамилия, длъжност, месторабота)

(Ако нямате научен ръководител, пишете НЯМА)

Cake

“Направи програмирането лесно като детска игра[1]”

„Cake“ е специално изработен мултипарадигмен компютърен език написан на Java и C++. Основните му цели са да предостави лесно въведение в програмирането на всеки, който се интересува, дори без никакъв предишен опит, докато в същото време езикът да бъде достатъчно мощен за да предостави възможност за създаване на сложни проекти. Също така, „Cake“ включва собствено IDE(среда за разработка на софтуер), написана на C#, позволяваща пълна независимост и не изискваща допълнителни познания.

Докато от една страна “Cake” е насочен към професионалисти с години опит, то от друга е фокусиран върху ученици заинтересовани в програмирането без никакъв. Целта му е да даде възможност на начинаещи да получат основите в програмирането без никакви начални познания. Също така, „Cake“ ще е лесен за работа, мощен практичен, което значи, че той ще е перфектен език за програмиране за преподаване в училищата.

„Cake“ позволява на напредналите разработчици да се справят с обемни проекти със сложна йерархия. Също така, “Cake” дава преднина в програмирането все още бидейки ученик, така че този език дава възможност на бъдещи разработчици да работят със същия език за програмиране, който са учили в училище без нужда от преквалификация на друг. По този начин “Cake” улеснява работата на програмистите с помощта на комплексни концепции и лесно управляеми фрагменти в програмирането.

[1] букв. от англ. „Направи програмирането лесно като парче торта“

Cake

Make programming easy as a piece of cake

“Cake” is a custom-engineered Java based and multi-paradigm programming language. Its main goals are to provide an easy startup to everyone interested in programming without having any experience and at the same time to be powerful enough to provide the opportunity to build sophisticated projects. Moreover, it features its own IDE to be fully independent and not to require any additional software or knowledge.

On the one hand, “Cake” is targeted to the professionals with years of experience and also students interested in programming with no programming background. Its aim is to provide the beginners with basic knowledge and a startup in coding from the ground up. Furthermore, “Cake” will be easy to work with, powerful and practical which will be the perfect programming language to be taught at schools.

On the other hand, “Cake” also allows the advanced developers to be able to cope with bulky projects with a big hierarchy. Not only that, Cake gives the head-start in programming while being still a student, so it gives the opportunity to the future developers to work with the same programming language that they have been taught at school without the need of prequalifying in different ones. Thus, “Cake” eases the programmers to do so by introducing the complex concepts in programming in manageable pieces.

1. Увод

Проектът „Cake“ има за цел разработката на нов програмен език, имащ като основа мотото „Make programming a piece of cake. “ (букв. „Да направим програмирането като парче торта“, да го направим лесно и разбираемо за начинаещи). Въпреки тази идея, която насочва към групата на по-младите, които предстои да навлязат в тази сфера, запазена е и функционалността на съвременните езици като са имлементирани нови структури, йерархии, конструкции и технологии, които позволяват изключително удобно използване както от начинаещи, така и от професионалисти.

Положени са максимални усилия да запазим мултипарадигмената концепция на езика, а именно обектно-ориентираната и функционалната природа. По този начин тласкаме синтаксиса към ориентировка към бъдещето и лесните поддръжка и дебъгване(2. Парадигми).

Поради своята имплементация и възможности езика не може да бъде определен нито като високо, нито като ниско ниво, тъй като има възможност както за изключително висока абстракция, така и за директно управление на паметта, което се отключва чрез специален атрибут(част от синтаксиса), но по своя синтаксис „Cake“ прилича повече на езиците от високо ниво по своята четимост и отношение към говоримите езици(3. Управление на паметта).

Компилатора както и IDE-то идват с някои готови програми, от които потребителя може да се самообучава. Тези примери са достатъчно добре документирани(на английски език), за да има възможност да се прочете кода, след което да се пусне програмата и да се изпробва функционалността (4. Примери).

Синтаксисът на езика има значителни разлики с много от съвременните езици, но има и забележими прилики, което обуславя бързото научаване какъвто и опит да има даден човек с програмирането и различните парадигми. Поради своята мултипарадигмена природа има свобода по отношение на дефиницията на някои структури(5. Синтаксис и граматика).

„Cake“ идва със сравнително малък, но мощен набор от основни типове данни. Благодарение на вградената библиотека (6. Типове данни).

Като всеки език е необходимо „Cake“ дефинира своите изисквания за идентификаторите, които ще се използват като имена на променливи, методи, типове и т.н.(6. Идентификатори)

За да се предостави възможност за разработване на големи проекти чрез „Cake“ е необходимо да се оформи ясна йерархия на програмния код както и да се спомогне low coupling(ниската зависимост между програмния код) и high cohesion(високата модуларност на програмния код, която го прави и по-четим) (4. Йерархична Структура)

В проекта е включен и Integrated Development Environment-а(IDE), който има оцветяване на ключовите думи и изпълнение на програмата (9.Integrated Development Environment).

Компилатора и IDE-то са изключително леки приложения, всеки съвременен компютър, който може да работи с Java(която е изискване за работа) може да работи и с компилатора.

Технологии, които са използвани в разработката на проекта са Java и C++. (10.Технологии)

2. Парадигми

Основните концепции, към които се придържа „Cake“ са обектно-ориентираното(ООП) и функционалното програмиране(ФП).

ООП определя специфичното групиране на данните подобно на реалния, материален свят. Тази парадигма предполага и третирането дори на функции като обекти, което улеснява работата с функции от по-висок ред и спомага ФП. ООП е доказана парадигма, която улеснява работата с големи проекти, именно по тази причина е използвана.

ФП се е оформила по-късно от ООП като концепция, но бързо наваксва и показва, че има огромен потенциал. ФП представя правилото с непроменящото се състояние на обектите, което е решено в „Cake“ чрез вътре-инстанционно променяне. Това, което функционалното програмиране дава на програмния език, е стабилност и яснота. Програмите написани с ФП имат интересното свойство да бъдат своеобразно доказателство за своето собствено функциониране, което ги прави изключително лесни за поправка и поддръжка.

Обединени тези 2 концепции оформят един изключително мощен програмен език с огромен потенциал. Още една технология, която се отваря е Read–eval–print loop(REPL), която улеснява работата и намалява времето и разходите за разработване на нов софтуер.

3. Управление на паметта

Тази дейност в езиците от високо ниво обикновено се извършва от Garbage Collector(GC). Това е автоматична система за почистване на паметта от неизползваните обекти.

Този процес при езиците от ниско ниво се извършва ръчно, което позволява по-голяма свобода при работа с паметта, но отваря и възможността за грешки.

В „Cake“ този проблем е решен чрез заключването на тази функционалност зад специален атрибут, чрез който само може да се отключи.

Това позволя на програмния език да бъде изключително свободен за разработка на приложения, при които производителността е ключова характеристика.

4. Примери

Примерите, които идват в комплект с компилатора все още са в процес на развитие, но те няма просто да представят особеностите на синтаксиса, но и ще бъдат достатъчно добре подбрани, за да въведат основни идеи от програмирането за начинаещите програмисти.

5. Синтаксис и граматика

Както всеки език така и „Cake“ дефинира собствен синтаксис, чрез който ще постига целите си.

I. Терминатори на команди(редове)

За разлика от много от доказалите се езици, „Cake“ не се нуждае от терминатори- какъвто е „ ; “ поради своята интересна структура. Това го прави по-четим и намалява броя символи нужни за постигането на цел. Не на последно място това свойство намалява и големината на получените файлове, при което се спестява памет.

II. White-space (празни места и табулации)

Както много съвременни езици, така и „Cake“ не зачита празните полета и табулациите като значима единица. Единствената невидима значима единица в езика е символа за край на ред.

III. Коментари

За да бъде модерен език „Cake“ трябва да предоставя начин за документиране на дейността на кода, а това се случва именно чрез коментари. Те биват 2 вида:
- коментари на един ред, дефинирани чрез „//“(двойни дясно-наклонени черти). Всеки символ след тази конфигурация бива игнориран от компилатора.
-коментари на няколко реда, дефинират чрез „ /* “ като отварящ и „ */ “ като затварящ показател. Всеки символ между тези двойки показатели бива игнориран от компилатора.

IV. Ключови думи

Положени бяха максимални усилия да се сведе до минимум използването на ключови думи, за да си избегне необходимостта от наизустяване на „речника“. Всички, които бяха избрани и използвани, са изключително близки до английски език с цел отново да се запази възможността всеки да работи с него(с оглед на факта,че от ¼ до ½ от населението на земята владее английски език поне на базово ниво).

V. Литерали(декларация)

Литералите могат да се разделят на няколко вида:

- Числови литерали

  В „Cake“ не се дискримира между цели и реални числа, ако не е експлицитно указано от разработчика.

  Примери за числови литерали са: 3, -5, 42.123, -172.5, 3.141592
- Символни литерали

  За символен литерал се счита всяка последователност от символи оградена с кавички „“. Примери: „Гошо“, „Cake“, „Th1$ is sti(l a $ymb0l l1teral“
- Булеви литерали

  Булевите литерали(още наричани логически) имат само 2 възможни стойности- **true** или **false**
- Литерал на празната идентичност

  Специален литерал указващ липса на данни. Всеки такъв литерал сочи към специален адрес в паметта, който се определя от всеки проект независимо. Всяка референция към този адрес в паметта може да бъде приравнен на празната идентичност

VI. Разделители(специални символи)

Това са символи изпълняващи специална роля.

Обикновено вървят по двойки – „()“ „[]“ „{}“ „ „“ “ също и „ : “ „ ; “ „ . “.

Всеки символ който НЕ може да бъде използван във построяването на идентификатор може да се счита за специален символ.

VII. Оператори

Разбира се като всеки език има готовите математически операции – събиране(+), изваждане(-), умножение(*), деление(/), модулно деление(%).

В допълнение и логическите операции – конюнкция(&&), дизюнкция(||), отрицание(!), изключваща дизюнкция(^^), равнозначност(==) и разнозначност(!=)

Една особено полезна и важна особеност на езика е възможността за предефиниране на операции за всеки тип. Това се случва в специални групи от методи(group operations), където се дефинират методи, които чрез своята логика извършват операцията между двете(респективно едната при отрицанието) и се получава нова стойност.

VIII. Атрибути

Едно от свойствата, което отличава „Cake“ от останалите езици е системата на атрибутите, които се дефинират преди всеки тип. Атрибутите могат да имат различни ефекти- от кеширане на данните до отключването на дирекния достъп до паметта споменато по-горе. Атрибутите мога да бъдат определени като своеобразни метаданни(данни за данните)

IX. Декларация на променливи/функции(методи)

Не се дискриминира между тези две декларации поради факта, че всеки елемент може да бъде третиран като инстанция на определен тип(дори методите)

X. Декларация на групи

Групи е название на именувана съвкупност от методи. Това позволява по-лесното групиране на данните и функционалността. Декларирането се получава чрез ключовата дума „group“ последвана от името на групата. След това се поставя блок с дефинирани методи променливи и подгрупи. Групите са своеобразни ограничители на видимостта. Променливи могат да бъдат декларирани вътре в тях като не са видими извън нея. Трябва да се отбележи, че има имена на групи, които са запазени за специална имплементация(това не пречи в тях да има друга разработка, но ще наруши принципите на ООП)

XI. Декларация на типове

Типовете са ключова част от ООП природата на езика затова е представена нова ключова дума „class“. Декларацията е чрез използване на ключовата дума, последвана от идентификатора на клас и накрая блок с дефинирани променливи методи и групи.

Това е основата на всяко приложение за това е обърнато специално внимание всичко в тази структура да е по природа организирано, за по лесна работа и бъдеща поддръжка

6. Типове данни

Главната причина за малкия набор от основни типове е, че в основната библиотека на езика има много имплементации на чести проблеми, за да може всеки потребител да ги използва без да се налага да прибягва до базовите типове. Разбира се, ако се налага да се използва алгоритъм за специфично решение, програмиста е свободен да избира своята имплементация или да направи своя.

I. Основни типове данни(примитивни типове данни)

a. Числови литерали

- Възможни операции

  Възможни операции да всички математически и логически действия, както и превръщанията в другите базови типове.

- Множество на допустимите стойности

  Допустими са всички рационални стойности(може да се представи като отношение между други две)

- Вариации

  Има 2 вариации, които позволяват по-лесно използване: цели числа и рационални числа. Това позволява по-голяма свобода на имплементацията за нуждите на потребителя.

b. Символни литерали

- Възможни операции

  Възможните операции са дефинирани от базовата имплементация на символния литерал. Както и стандартната конкатенация.

- Множество на допустимите стойности

  По дефиницията на символните литерали.

c. Булеви литерали

- Възможни операции

  Всички дефинирани логически операции.

- Множество на допустимите стойности

  Това множество има само 2 елемента- true и false

II. Комплексни типове данни

В други езици наричани „референтни типове“, това са типове данни съставени от други стойности, които построяват идентичността на даден тип и неговата инстанция. Поради своята функционална наклонност избрахме тези типове да могат да имат само възможност за четене на техните публични полета(освен чрез директна промяна директно на адресса в

паметта). Единствено инстанцията(чрез вътрешен механизъм) може да променя своите състояния. Всъщност в „Cake“ всеки тип е референтен, за да можем да запазим възможността за директно управление на паметта.

a. Наследяване(inheritance)

Наследяването е съществено свойство на ООП програмирането, което представлява пренасяне на признаци на един тип на друг, ефективно ставайки родител и съответно подтип.

b. Превръщания(cast-ване)

Превръщанията в „Cake“ се случват по уникален начин, а именно чрез специална група(group casting) в която се дефинират специални методи за представянето на инстанцията като друг тип данни или приемането на други данни и представянето им като инстанция на типа.

III. Празната идентичност

Това е специална референтна стойност посочваща адрес в паметта приет за празна идентичност. Този адрес се изчислява за всеки проект поотделно. В зависимост от имплементацията стойността на този адрес може да се променя, но това не е достъпно без необходимия атрибут(такава дефиниция може да има единствено една във всеки проект, в противен случай бива образувана грешка).

IV. Типови променливи

Тази част от езика все още не е готова, но в бъдеще предоставя възможност за Generics, за по удобна работа със Списъци, Речници и различни колекции.

7. Идентификатори

Като идентификатор се определя всяка последователност от букви, цифри, долни тирета, както и обикновени тирета, незапочващи с цифра. Идентификатори могат да се използват като имена на променливи, типове, методи и т.н.

I. Пълни имена(fully-qualified name)

Като пълно име се счита последователността от структури, които водят до даден идентификатор, разделени с точка/и.

8. Йерархични структури

Най-общата структура е „Cake“(да не се бърка с името на програмния език). Тя съдържа целия обем програмен код, както и дефиницията на функциите, които други „Cake“-ове могат да използват(декларация на API).

Следваща по ранг йерархична структура е bundle, представлява папка, която отново дефинира какво външни за нея структури могат да използват, може да съдържа други bundle-ли или програмен код

. И най-малката структура, която съдържа в себе си сорс код е „Slice“, тя дефинира най-често един тип, който може да се използва от друг програмен код.

I. Използване на Slice-ове от други структури

Чрез ключовата дума „use“ последвана от пълното име на използвания тип.

9. IDE

10. Използвани технологии

Избора на Java и C++ не е тривиален, поради функционалността, която предоставят функционалност полезна за проекта. Използвани са различни среди за разработка, но с предимство бяха платформите с отворен код(open-source). Използвана е библиотека Junit5, която се използва за тестване на програмния код, и Gradle, система за улеснено компилиране на проекта.

11. Заключение

Поставената цел пред проекта бе да се разработи програмен език, удобен за използване в големи проекти(модерен), но същи и достатъчно разбираем за начинаещи. Проекта постигна своята цел, но като всеки проект с отворен код ще претърпи промени, за да постига тези цели по-лесно.

Авторите разработиха този проект без никаква основа, на която да стъпят, и имат виждане за неговото бъдещо развитие като общо-популярен програмен език от ранга на Java, C#, C++ и др.

12. Използвана литература

I. Въведение в програмирането с Java, автор Светлин Наков и колектив https://www.introprogramming.info/intro-java-book/

II. A complete guide to Programming in C++, автор Ulla Kirch-Prinz, Peter Prinz http://www.lmpt.univ-tours.fr/~volkov/C++.pdf

III. [Let's Make a Programming Language], обучителни епизоди, автор pogosticks29

IV. OOP – Learn Object Oriented Thinking & Programming, Bruckner Publishing

V. Java Design Patterns, автор Rohit Joshi http://enos.itcollege.ee/~jpoial/java/naited/Java-Design-Patterns.pdf