הצגת שפה -GOSU

אחינועם מונסונגו 206552382

שולמית נהון 323782852

.GS

Programming language

רקע ומטרת השפה

Guidwire Software היא חברה שמפתחת תוכנה עבור סוכנויות ביטוח, השפה פותחה כדי לשלב ולהגדיר את הלוגיקה העסקית של החברה בצורה פשוטה.

GOSU היא שפה חדשה יחסית, מבוססת JAVA שמרחיבה את הפונקציונליות של JAVA ומנסה להפוך דברים ליותר פשוטים, ועומדת בסטנדרט של ECMAScript.

היסטוריה

- **2002** - תחילת פיתוח השפה שנקראה בהתחלה GScript בידי החברה 'Guidewire Software'
- **2010** - 8 שנים לאחר התחלת הפיתוח, שוחררה בנובמבר הגרסא הציבורית 0.7 בשם החדש **GOSU**, ובדצמבר שוחררה הגרסא 0.8 כקוד פתוח
- **2011** - עודכנה הגרסא 0.8.6 עם אפשרות לעבודה עם XML
- **2016** - שוחררה גרסא 1.10 שכוללת תוסף סביבת העבודה Intellij IDEA ל- GOSU

מאפייני השפה

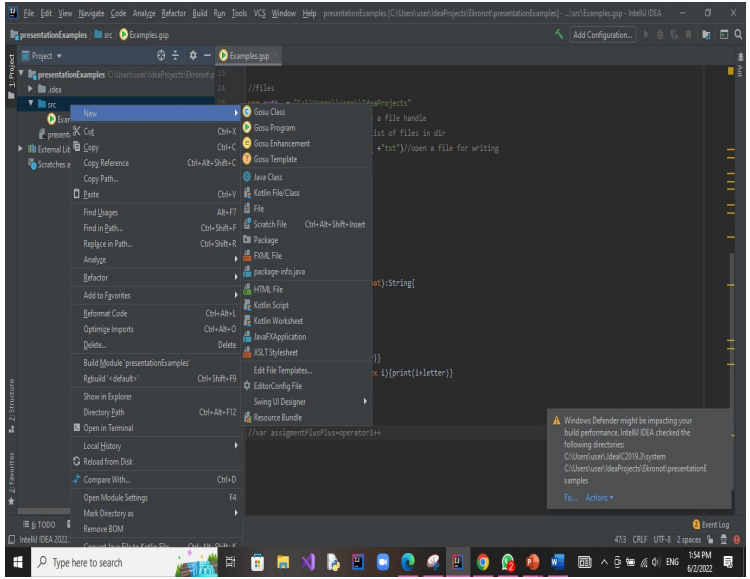
GOSU היא שפה עילית אימפרטיבית מונחית עצמים, דומה מאוד לJAVA.
תומכת ב:

- תכנות מונחה עצמים.
- עבודה בתיאום עם קבצי JAVA
- זיהוי אוטומטי של טיפוסים
- ביטויי למבדה
- עובדת על JVM -תכונה המסייעת לניידות של השפה

```
var listOfWords ={"Programming","is","fun!"}
var someWords = listOfWords.where(\str -> str.length()>3)
```

Object Oriented

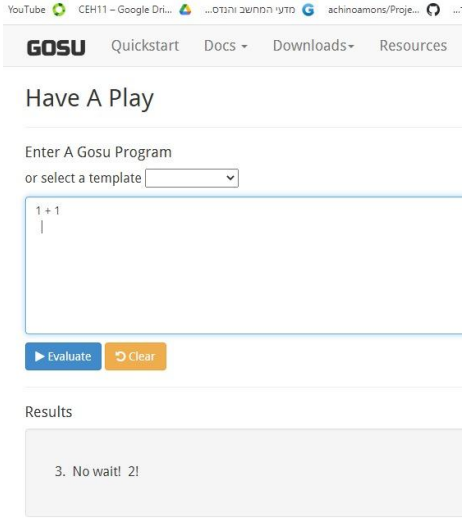
- **אבסטרקציה**-הפשטה. על ידי מחלקות ופונקציות
- **ירושה**-ניתן לרשת רק ממחלקה אחת ,ניתן לרשת מכמה ממשקים.
- **פולימורפיזם**-רב צורתיות,אפשרי בשפה. (למשל בצורה הקלאסית של OOP- כלומר ירושה)
- **Information Hidding**-כמו בJAVA. ברירת המחדל היא package-private כלומר נגישות לכל חברי החבילה הנוכחית.



סביבת עבודה

ישנן שלוש אופציות לכתוב קוד בשפת GOSU :

- בעזרת סביבת הפיתוח המיועדת במיוחד לשפה.
- בעזרת סביבת הפיתוח IntelliJ, ובעזרת פלאגין המיועד לשפה.
- בעזרת העורך האינטרנטי, המיועד לתכניות קצרות ולא מסובכות.



Readability

יתרונות

השפה נוחה לקריאה והמבנה ברור לכל מתכנת.
GOSU הינה מבוססת JAVA ולכן מתאפיינת ב:

- Control Statenent-נוכחותם של מבני בקרה ידועים כדוגמת while:
- Syntax consideration-שיקולי תחביר כמו פונקציות,מילים שמורות המקלות על ההבנה
- Orthogonality (אין מצביעים) בדומה לJAVA

חסרונות

- העדר ; בסוף פקודה שעלול לבלבל ולהקשות כאשר ישנו קוד ארוך
- סוגי הנתונים אינם מוכרחים להיות מוצהרים מראש- השימוש ב VAR בהגדרת משתנה עלול לבלבל.

```
function func1( ):String{
    return  "Hello World!"
}
```

```
var x=func1()
```

Writability

יתרונות

- תמיכה באבסטרקציה -הפשטה:
  - הפשטת תהליך- פונקציות
  - הפשטה מבנית-מחלקות,ירושה,הכמסה
- Expressivity-יכולת הביטוי של השפה. דרכים נוחות לבצע פעולות,יש פחות מגבלות תחביר.
- אין צורך להצהיר מראש על סוגי הנתונים.

חסרונות

++ / -- - Just like the Java operators, except they cannot be used within another statement

Reliability

יתרונות

- Garbage Collector -פותר את בעיות המצביעים: זליגת זיכרון ומצביע מתנדנד.
- Soppurts Exception handling

```
public Object getValue( Object ctx )
{
 try
 {
  return _getter.invoke( _type, null );
 }
 catch( Exception e )
 {
  throw GosuExceptionUtil.forceThrow( e );
 }
}
```

חסרונות

- אין צורך להצהיר מראש על סוגי הנתונים-מקטין אמינות.

Cost

- קל להכשיר מתכנתים
- IDE להורדה בחינם
- Open Source
- JVM הינו חינמי (יפורט בהרחבה בהמשך)

Portability

GOSU עובדת על JVM, עובדה התורמת לניידות של השפה

JVM-Java Virtual Machine

אחד האלמנטים שהופכים את GOSU לשפה חשובה הוא ה JVM . כאשר אנחנו מהדרים קוד GOSU המהדר איננו יוצר קוד בשפת מכונה, אלא יוצר קוד ביניים- byte code. לפני שהתוכנית שאנחנו כותבים רצה על מחשב היעד, היא עוברת קומפילציה נוספת כדי להתאים לשפת המכונה של אותו מעבד.

- בצורה כזו תוכנית שנכתבה ב GOSU יכולה לרוץ על כל מערכת הפעלה שתומכת ב-Java ללא שינויים ואף ללא צורך בהידור מחדש.

מצביעים

- בשפת GOSU אין מצביעים,השימוש הוא ב references בדומה לJAVA.
- סיבה עיקרית : להגביר את האמינות והבטיחות של השפה, הורדת המצביעים מונעת ממתכנתים לעקוף את מנגנון האבטחה של השפה ומונעת בעיות שקיימות במצביעים.
- ה- GC משחרר את הקצאת הזיכרון באופן אוטומטי.

```
var compilationEngine: CompilationEngine  = new CompilationEngine(currentJackFile,fileOut,tokensFileOut)
```

Garbage Collector

- השפה תומכת במנגנון garbage collector הפועל באופן אוטומטי והאחריות לשחרור הזיכרון אינה מוטלת על המתכנת.
- המנגנון עובד בשיטת Mark & Compact - ריצה על כל ההצבעות שנמצאות בשימוש אל אובייקטים הנמצאים בזיכרון באזור הערימה (שהוקצו דינמית), וסימון של כל האובייקטים הנגישים מהמחסנית כאובייקטים בשימוש תוך כיווצם לתחילת הערימה. לבסוף, כל האובייקטים שאינם מסומנים משוחררים.

טווח הכרה - Scope

- static scoping-טווח ההכרה של משתנה הוא לפי עץ המדרג הסטטי ונקבע לפי הגדרתו בהרכיה של מבנה התכנית.
- אין תמיכה בקינון פונקציות

תחביר השפה

המבנה הבסיסי הוא מבנה אימפרטיבי הכולל:

1. פקודות
2. הגדרות משתנים
3. בקרת זרימה (לולאות ותנאים כגון if,while)
4. הגדרת ושימוש בפונקציות
5. הגדרת מחלקות

בניגוד לרוב השפות מונחות העצמים, בשפת Gosu ניתן לכתוב פקודות מחוץ למחלקה או פונקציה.

מנגנון הטיפוסים

- **static typing**-סוג המשתנה נקבע בזמן קומפילציה.
ניתן לא להגדיר בפירוש את הסוג כל עוד הקומפיילר יכול להסיק את הטיפוס לפי ההקשר.

```
var a=1.5 //float
var b: int //integer
var c="string"//string
```

- **מנגנון חלש**-מאפשר המרה מרומזת .השפה מבצעת המרה אוטומטית מרומזת (כשהדבר אפשרי).

```
var implicit=6
var str= "hi"+x //hi6
```

טיפוסי נתונים ומימושם

- טיפוסים בסיסיים
- טיפוסים מורכבים לדוג' Enum,dictionary
- ניתן להגדיר טיפוסים ואובייקטים

```
enum Letter {A,B}
var dictionary={ Letter.A->"a",Letter.B->"b" }
```

מבני בקרה וזרימה

בשפה ישנם שני סוגי תנאים: if else ו - switch-case.

בנוסף שלושה סוגי לולאות, גם הן הקלאסיות: for, while ו - do-while

ללולאת ה-for ישנו מבנה שונה, כמתואר להלן:

```
for (number in 0..9){print(number)}
for(letter in "we love GOSU" index i){print(i+letter)}
```

פונקציות

```
function IloveGosu(i:int,love:float):String{
 return "I Love GOSU"
}
```

מילה שמורה בתחילת פונקציה מכל סוג שהוא

סוג הפרמטר

שם הפרמטר

הערך המוחזר

- בשונה מJAVA ניתן לרשום פונקציות שלא בתוך מחלקות בעזרת GOSU PROGRAM

העברת ערך לפונקציה

- העברת ערך של משתנים בסיסיים מתבצעת בשיטת **pass-by-value**, כלומר נוצר העתק של המשתנה ולא נשלח המשתנה עצמו.
- משתנים מורכבים (מחלקות), עוברים בשיטת **pass-by-reference**, כלומר הפרמטר מקבל גישה למשתנה עצמו שנשלח לו, וכל שינוי בפרמטר, הוא בעצם שינוי במשתנה המקורי.
- **ניתן לשים ערכי ברירת מחדל בפרמטרים הפורמליים**

```
// A public function
function printNames( prefix : String = "> ") {
  for( n in _names ) {
    print( prefix + n )
  }
}
```

```
var c = new SampleClass({"joe", "john", "jack"})
c.printNames(:prefix = "* ")
```

סדר הפרמטרים

- **by position**-סדר הפרמטרים בשורת הקריאה צריך להיות זהה לסדר הפרמטרים בהצהרה, מכיון שהמיקום הוא המזהה של המשתנה כמשוייך לפרמטר כלשהו.
- **by keyword**-משתמשים בשם הפורמלי על מנת לקשר אליו פרמטר אקטואלי

```
// A public function
function printNames( prefix : String = "> ") {
  for( n in _names ) {
    print( prefix + n )
  }
}
```

```
var c = new SampleClass({"joe", "john", "jack"})
c.printNames(:prefix = "* ")
```

Programming language

אופרטורים והשמות

Gosu supports the standard Java operators, with a few minor restrictions and some great bonuses:

- `++` / `--` - Just like the Java operators, except they cannot be used within another statement
- `==` - Tests for object equality, just like `.equals()`
- `===` - Tests for *instance* equality
- `<>` - The same as `!=`
- `<`, `>`, etc - Standard comparison semantics which also works on `java.lang.Comparable` objects

```
var operator1=0
var assigmentPlusPlus=operator1++
```

מקביליות

המקביליות של השפה בנויה ממודל המקביליות של שפת JAVA.

בכדי להתחיל תהליכון נשתמש בפונקציה Thread.start()
ובכדי להרדים את התהליך נשתמש בפונקציה Thread.sleep()

lazy evaluation

- **Lazy concurrent variables.** The `LazyVar` class (in `gw.util.concurrent`) implements what some people call a *lazy variable*. This means Gosu constructs it only the first time some code uses it. For example the following code is part of a class definition that defines the object instance. Only at run time at the first usage of it does Gosu run the Gosu block that (in this case) creates an `ArrayList`:

```
var _lazy = LazyVar.make( \-> new ArrayList<String>() )
```

Programming language

קבצים

```
//files
var path_ = "C:\\Users\\user\\IdeaProjects"
var dir = new File(path_)//create a file handle
var file_list = dir.list()//get list of files in dir
var output = new FileWriter(path_ +"txt")//open a file for writing
output.write("hello")
output.close()//must close!
```

מחרוזות

בנוסף לפקודות שיש לjava, בגלל השימוש הרב במחרוזות gosu הוסיפה עוד כמה פונקציות משלה:

- `repeat(n:int)` - שכפל המחרוזת n פעמים
- `()chomp` - אם יש, תוריד שורה חדשה עוקבת מסוף המחרוזת
- `()chop` - תוריד את התו האחרון מהמחרוזת
- `elide(len:int)` - תגדיר אורך קבוע למחרוזת, ואת ה3 תווים האחרונים אחרי צמצום האורך, תחליף ב: '...'
- `rightPad(w:int), leftPad(w:int), center(w:int)` - עצב את המחרוזת עם רווחים נוספים
- `()notBlank` - תחזיר אמת אם המחרוזת לא null ומכילה לפחות תו אחד שהוא לא רווח

טיפול-Exceptions בשגיאות

- נעשה דרך מבנה ה-try...catch

Syntax

```
try
  <try statements>
[catch( exception )
  <catch statements>]
[finally
  <finally statements>]
```

Null-Safty

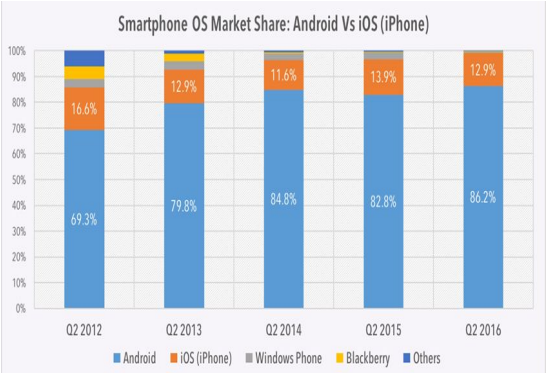
- דרך למניעת קבלת שגיאה במקרה של null לחלק מהטיפוסים.

```
var str1:String
var nullSafty=str1?.substring(3)
```

יתרונות וחסרונות

יתרונות

- שפה פרקטית וקלה ללימוד
- יש זיהוי אוטומטי של טיפוסים - קוד נקי
- סביבת עבודה נוחה
- בעלת יכולת פעולה הדדית ותאימות עם JAVA וספריותיה בצורה חלקה וכמעט שקופה



חסרונות

- שפה פחות מוכרת לכן אין הרבה תמיכה ומקורות הידע מוגבלים
- כרגע אינה מתאימה לפיתוח אפליקציות אנדרואיד.

https://www.answersjet.com/2021/08/gosu-programming-language-history-features-applictions-why-should-learn-gosu-lang.html

The only way to learn a new programming language is by writing programs in it.

Dennis Ritchie

quotefancy

.Gs
Programming language

תודה רבה!

achinoamons@gmail.com
shulamitna26@gmail.com