PROGRAMLAMA DİLLERİ

- Program
- Yazılım Geliştirme Süreci
- Programlama Dilleri Tarihçesi
- Kod Sözdizimi
- Nesneye Yönelik Programlama
- Tür Kontrolü
- Alt Programlar
- Programlama Dilleri
- Programlama Dillerinin Önemi
- Dilleri Sınıflandırılması
- Anlambilim
- BNF Notasyonu
- Kontrol Deyimleri

Muhammet Baykara
mbaykara@firat.edu.tr

Program Nedir?

- **Program:** Bilgisayarın bir işlevi yapması için tasarlanmış komutlar zinciridir.
- Ardışık simgeler dizisi.
- **Programlama Dili:** Bir makinenin davranışını kontrol etmek için kullanılabilen ve program üretimi için kullanılan yapay dil.

Yazılım Geliştirme Süreci

- **Gereksinim Analizi:** Yazılım projesinin ilk etabında projenin ihtiyaç duyduğu ana modüller analiz edilmeli, (software requirements specification) proje amaçları ve hedefleri detaylandırılmalıdır.
- **Tasarım:** Oluşturmak istenilen proje web tabanlı, mobil veya masaüstü olabilir. Bu doğrultuda yapacağınız tasarımın bu platformlara veya cihazlara uygun olması gerekmektedir.
- **Kodlama:** Güçlü bir yazılım mimarisi ile çalışılmalı ve sonradan çıkabilecek tüm isteklere kolaylıkla cevap verebilecek şeklide kodlama yapılmalıdır.

Yazılım Geliştirme Süreci

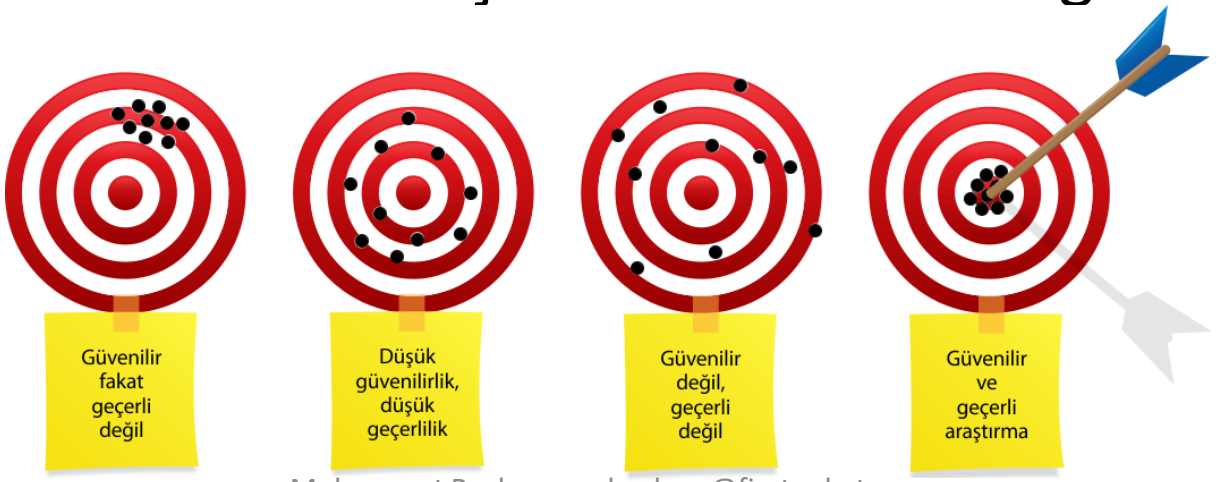
- **Sertifikasyon (Test):** Önceden belirlenen gereksinimlerin karşılanıp, karşılanmadığı doğru çıktıyı üretip, üretmediği testleri yapılmalıdır. Kalite denetimi yapılarak kullanıcıya sunulmasıdır.
- **Bakım:** Teslimat sonrasında çıkan sorunları gidermek ve performansı artırmak için yazılım uygulamasını değiştirmek ve güncellemektir. Bu değişimler basit kodlama hatası şeklinde olabileceği gibi tasarımdan kaynaklanan hataların giderilmesi şeklinde de olabilir.

Programlama Dilinin Önemi

- **1. Yazılım Güvenilir Olmalıdır**
- Güvenli yazılımlar kullanmak programcılar için önemlidir. Kullanılan programların güvenli sayılabilmesi için bazı kalite ölçütlerine sahip olması gerekir. Bu ölçütler;
- **a. Yazılabilirlik:** Problemin gerektirdiği veri giriş çıkış, akış denetimi gibi unsurların programlama dili tarafından karşılanabilir olması ile ilgili bir kavramdır.
- **b. Okunabilirlik:** Okunabilirlik programın mantığını takip etmek, sınamak ve hata ayıklayabilmek ile ilgili bir kavramdır.
- **c. Sıra dışı durumları karşılayabilme:** Programlama dili hatalı giriş, aritmetik taşma gibi durumlara hazırlıklı olmalı bu durumlara uygun çözümleri içermelidir.

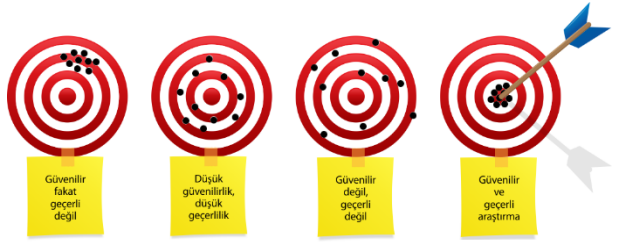
Programlama Dilinin Önemi

- **2. Yazılım Bakıma Elverişli Olmalıdır**
- Daha önceden geliştirilmiş bir yazılımın hatalarının ayıklanması ve giderilmesi, yeni modüller eklenmesi ve çıkarılması kullanıcı ihtiyaçları doğrultusunda yazılımın geliştirilmesi yazılım bakımının temel unsurlarıdır.
- Yazılımın okunabilirliğinin yanında değiştirilebilirliğinin de yüksek olması o programlama dilinin bakıma elverişli olması anlamına gelmektedir.



Programlama Dilinin Önemi

- **3. Yazılım Verimli Çalışmalıdır.**
- Geliştirilen yazılımların verimliliği kullanılan programlama dilline doğrudan bağlıdır. Verimlilik esas olarak programlama dilinin tasarımı sırasında ele alınan bir konudur.
- İlk dönem programlama dillerinde verimlilik programlama dilinin mevcut donanımı etkin ve hızlı bir şekilde kullanması ile ölçülmekteydi.
- Bu gün bu anlayış değişmiş ve programlama dilinin verimliliği işletim hızı, bellek gereksinimi, geliştirme ve bakım sırasında harcanacak zaman gibi kavramlarla ölçülen bir değer olmuştur.



Okunabilirlik Temel Teorem

- ***Kod, başkaları tarafından en az zamanda anlaşılsın diye yazılmalıdır.***

- ***Az kod her zaman daha iyi midir?***

Genellikle, daha az kod yazmak daha iyidir ama her zaman az olan iyi olacak diye bir şey yok.

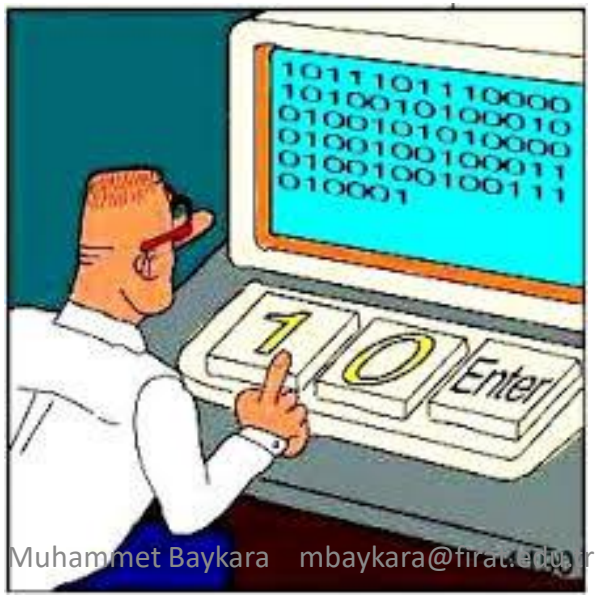
```
1   // ilki
2   assert((!(bucket = FindBucket(key))) || !bucket->IsOccupied());
3
4   // ikincisi
5   bucket = FindBucket(key);
6   if (bucket != NULL) assert(!bucket->IsOccupied());
```

İyi kaynak kodu göze kolay gelmelidir! Estetik

```
// ilki
class StatsKeeper {
public:
// A class for keeping track of a series of doubles
void Add(double d); // and methods for quick statistics about them
private: int count; /* how many so far
*/ public:
      double Average();
private: double minimum;
list<double>
  past_items
  ;double maximum;
};

// ikincisi
// A class for keeping track of a series of doubles
// and methods for quick statistics about them.
class StatsKeeper {
  public:
    void Add(double d);
    double Average();

  private:
    list<double> past_items;
    int count; // how many so far

    double minimum;
    double maximum;
};
```

Programlama Dillerinin Tarihçesi

- İlk programlar fiziksel olarak yazılıyordu. Daha sonra fiziksel programlama yerini elektrik sinyaline bıraktı. Artık, kurulan elektronik devrelere düşük ya da yüksel voltajda akım gönderilerek bilgisayarın davranışı belirlenmeye başlandı. Yüksel voltaj 1, düşük voltaj 0 sayılarını ifade ediyordu.
- Böylelikle bugün de kullanılan makine dilinin ortaya çıkması için ilk adımlar atılmış oldu.



Programlama Dillerinin Tarihçesi

- Ancak bu şekilde programlar yazmak, sistemi oluşturan elektronik devrelerin her program için baştan kurulmasını gerektiriyordu. Bundan dolayı programlar bazı kavramlar çerçevesinde yazılmaya başlandı.
- Öncelikle bilgisayar donanımı her program için baştan kurulmamalı, bunun yerine basit bir donanımın üzerine yazılan komutlar kullanılmalıdır. Daha sonra, programlar tek bir komutlar zinciri yerine, küçük parçalar halinde yazılmalıdır.
- Bu parçaların programın içinde defalarca kullanılabilmesi yordam (subroutine) kavramını ortaya çıkarmıştır. Bu modelin kullanılması ise mantıksal karşılaştırmaları, döngülerin kullanılmasını ve yazılan kodlar tekrar kullanıldığı için kütüphane (library) mantığını ortaya çıkarmıştır.

Programlama Dillerinin Tarihçesi

- 1957 yılında IBM, düşük seviye (makine diline yakın) bir programlama dili olan FORTRAN dilini ortaya çıkardı.
- FORTRAN ile beraber basit mantıksal karşılaştırmalar, döngüler, (true-false) lojik ve (integer, double) sayısal değişkenler kullanılmaya başlandı.
- 1959 yılında, bu programlama dilinin özelliklerini alıp, giriş çıkış (Input – Output IO) gibi yeni işlevler sağlayan COBOL dili ortaya çıktı.
- Daha sonra 1968 yılında, COBOL ve FORTRAN dillerinin en iyi özelliklerini alarak Pascal ortaya çıktı. Ayrıca Pascal dili, hafızadaki adresler üzerinde işlem yapmaya olanak veren işaretçi (pointer) kavramını beraberinde getirdi.

Programlama Dillerinin Tarihçesi

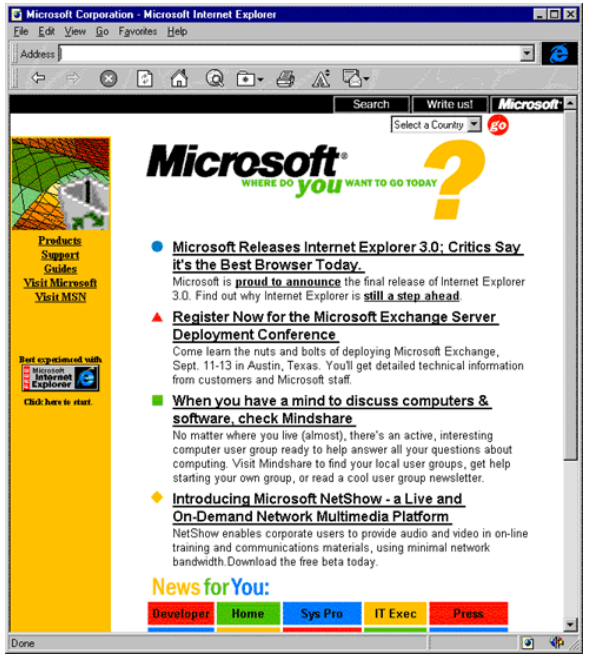
- 1972 yılında C, Pascal dilindeki birçok hatayı gidererek ortaya çıktı. C dili ilk defa Unix işletim sistemini yazmak için kullanılmaya başlanmıştır.
- C, orta seviye bir dil olması, kuvvetli giriş çıkış işlemleri sağlaması gibi birçok özelliği ile işletim sistemleri yazılmasında tercih edilmiştir.
- Bütün programlama dilleri birçok özelliğe sahip olmasına rağmen, modüler programlamanın birçok eksiğini gidermek amacıyla, yeni bir programlama modeli olan nesneye yönelik programlama - OOP (object oriented programming) ortaya çıkarıldı. C dilinin ve OOP modelinin tüm özellikleriyle C++dili oluşturdu

Programlama Dillerinin Tarihçesi

- C++ dilini, Sun Microsystems tarafından çıkartılan Java takip etti. Java dilinin kullanım alanları, nesneye yönelik bir programlama dili olması ve beraberinde getirdiği çöp toplama GC (garbage collection) gibi performans arttırıcı özellikleri ile büyük ölçüde genişledi.
- Microsoft, 2000 yılında .NET platformunu sunarak, otuzdan fazla programlama dilini aynı çatı altına topladı. VisualBasic.NET ve VisualC# .NET platformunu kullanan günümüzdeki en güçlü yüksek seviyeli programlama dilleri arasında yer almışlardır.

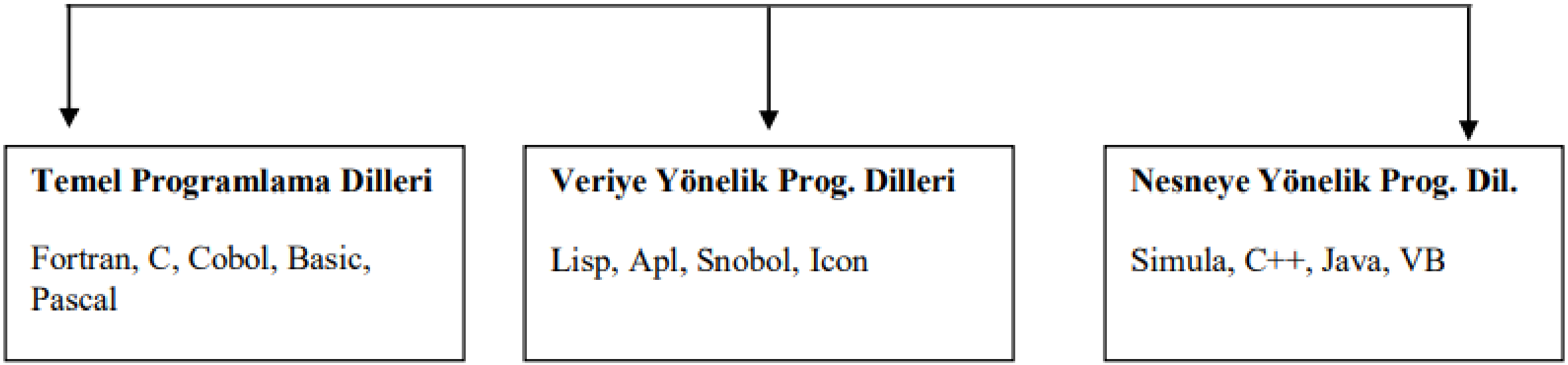
Programlama Dillerinin Tarihçesi

- C++ dilini, Sun Microsystems tarafından çıkartılan Java takip etti. Java dilinin kullanım alanları, nesneye yönelik bir programlama dili olması ve beraberinde getirdiği çöp toplama GC (garbage collection) gibi performans arttırıcı özellikleri ile büyük ölçüde genişledi.
- Microsoft, 2000 yılında .NET platformunu sunarak, otuzdan fazla programlama dilini aynı çatı altına topladı. VisualBasic.NET ve VisualC# .NET platformunu kullanan günümüzdeki en güçlü yüksek seviyeli programlama dilleri arasında yer almışlardır.



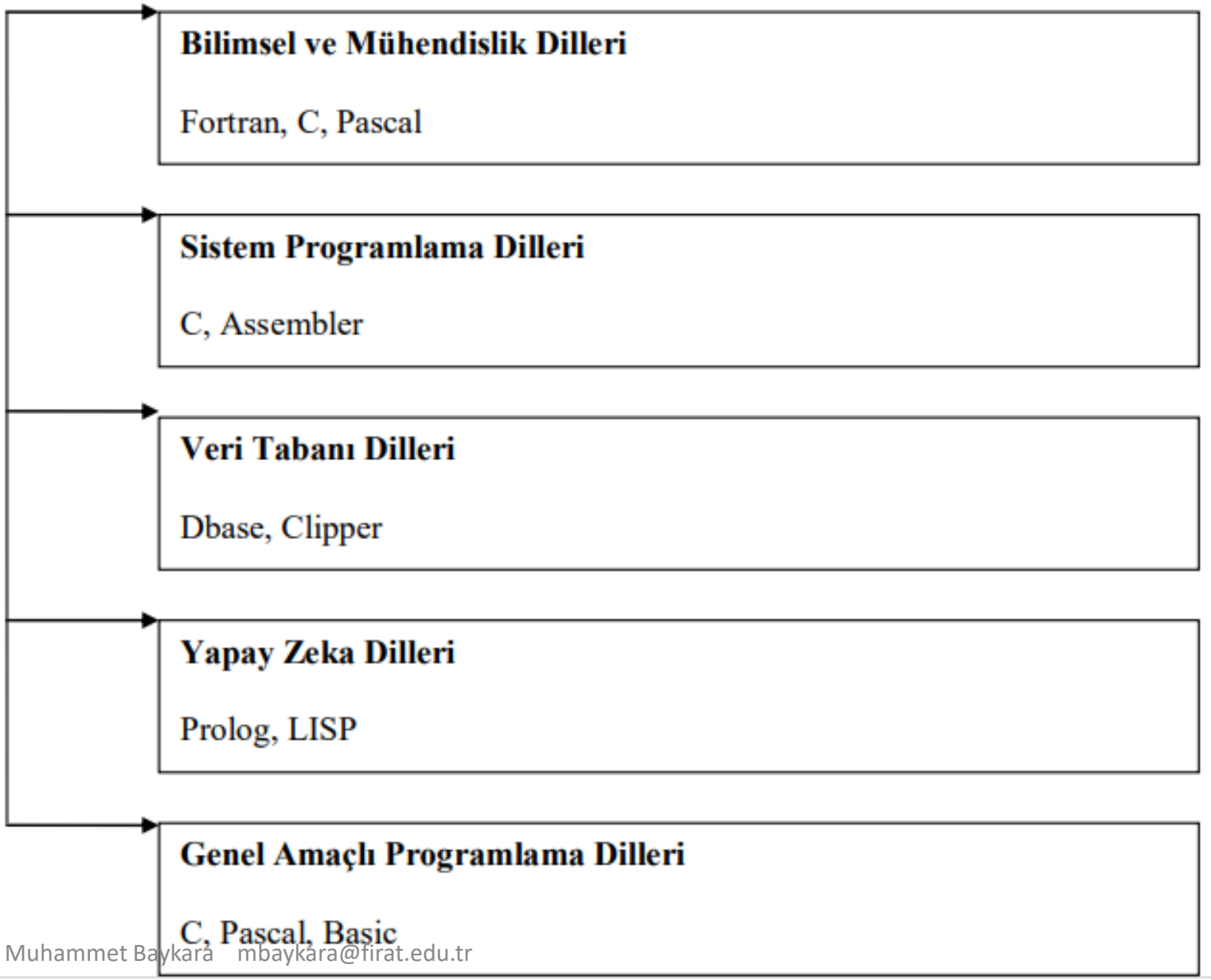
Programlama Dillerinin Sınıflandırılması

1- Genel Sınıflandırma



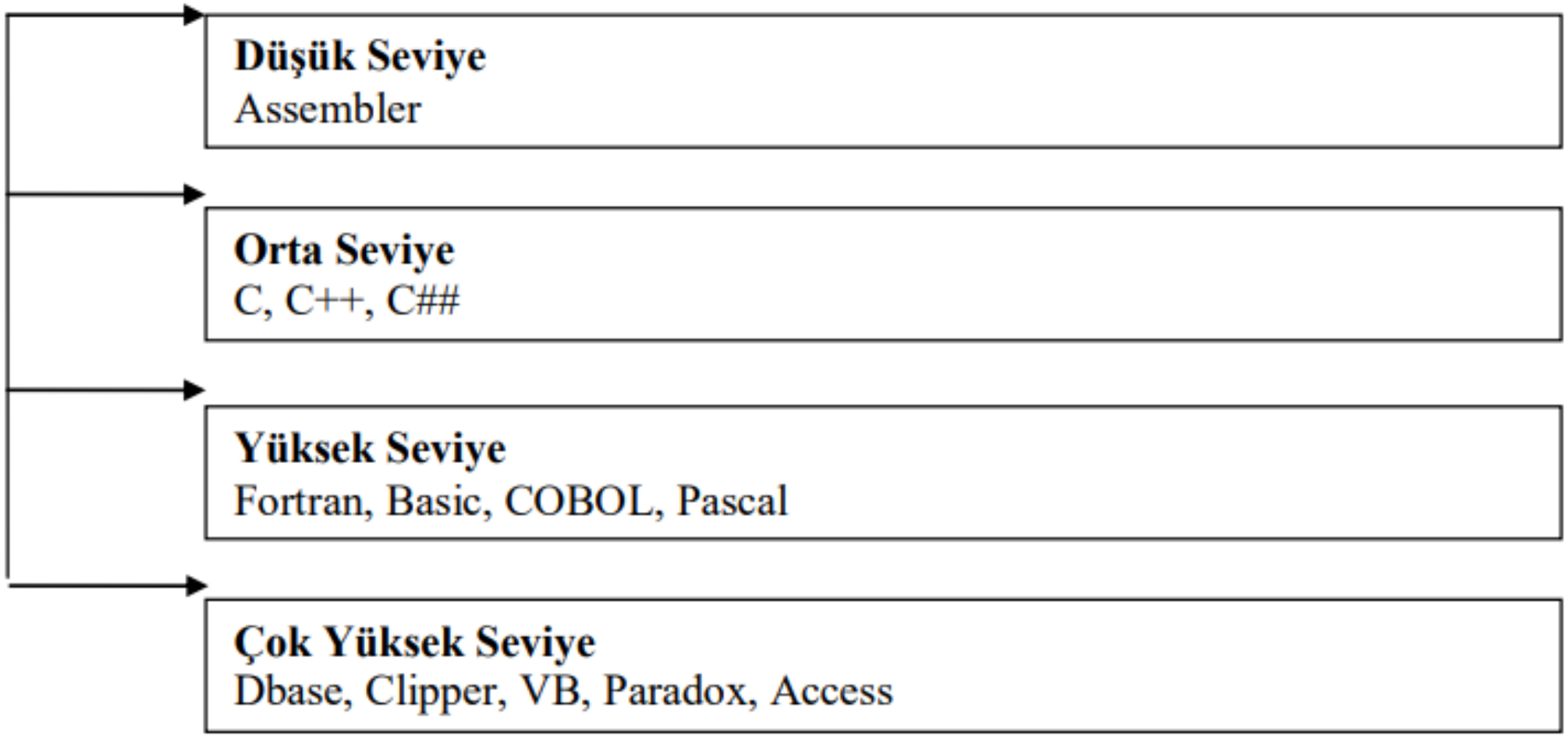
Programlama Dillerinin Sınıflandırılması

2- Uygulama Alanlarına Göre Sınıflandırma



Programlama Dillerinin Sınıflandırılması

3-Seviyelerine Göre Sınıflandırma

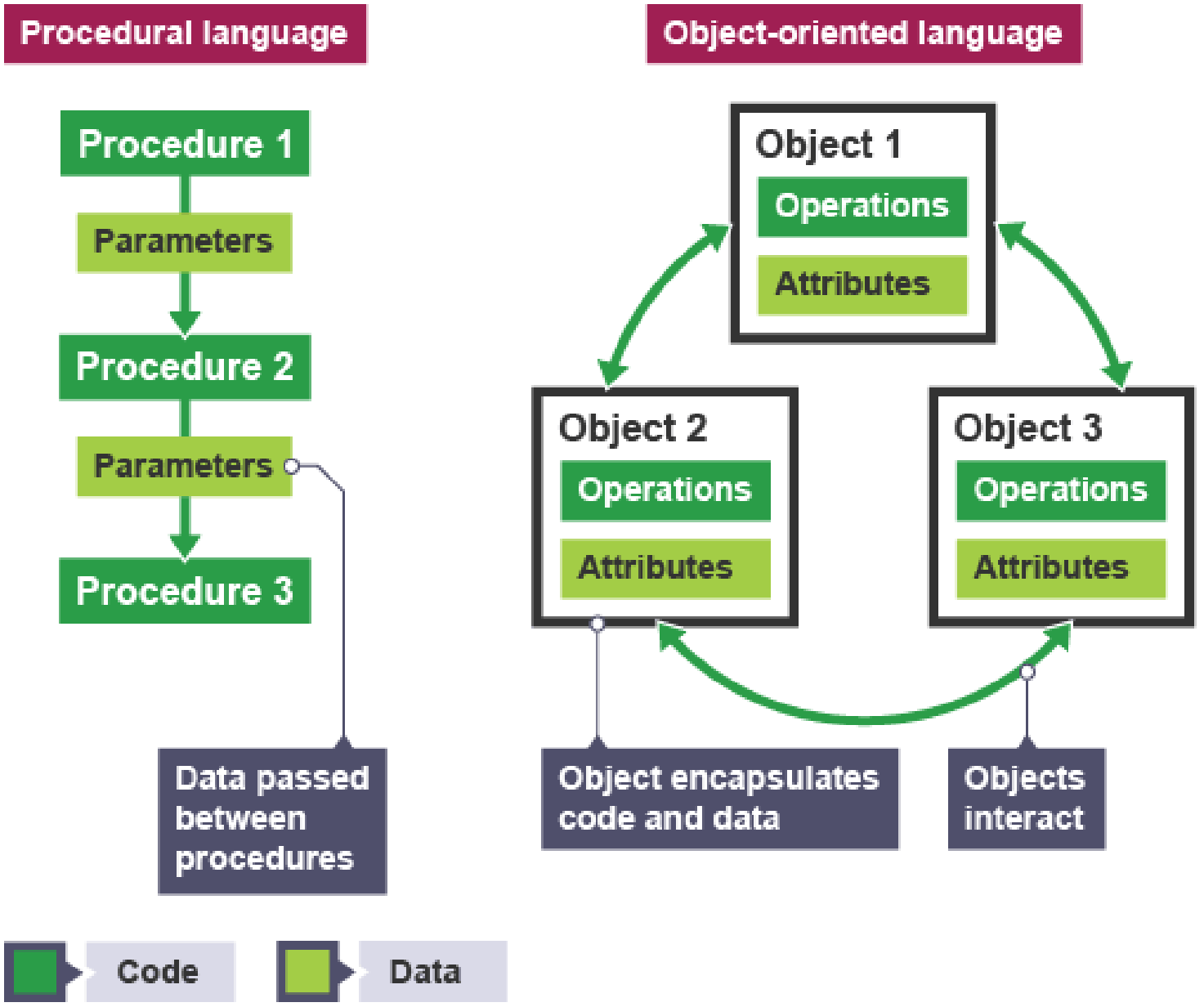


Nesneye Yönelik Programlama

- Nesneye yönelik programlama nesne kavramına dayanmaktadır. Burada nesne gerçek dünyada var olan veya programcı tarafından oluşturulmuş mantıksal bir varlıktır. Nesne, kendisini tanımlayan veriler ve bu veriler üzerinde yapılacak tüm işlemler ile bir bütün olarak düşünülür.
- **Veri Soyutlama (Data Abstraction) :** Kullanıcı tarafından yeni veri türlerini modelleyen sınıflar oluşturulmasıdır.
- **Kalıtım (Inheritance) :** Oluşturulan bu sınıfların genişleterek veya özelleştirilerek yeni sınıflar oluşturulmasıdır.
- **Çok Biçimlilik (Polymorphism) :** Aynı isimdeki işlemlerin değişik nesne grupları tarafından farklı algılanmasıdır.

Nesneye Yönelik Programlama Dilleri

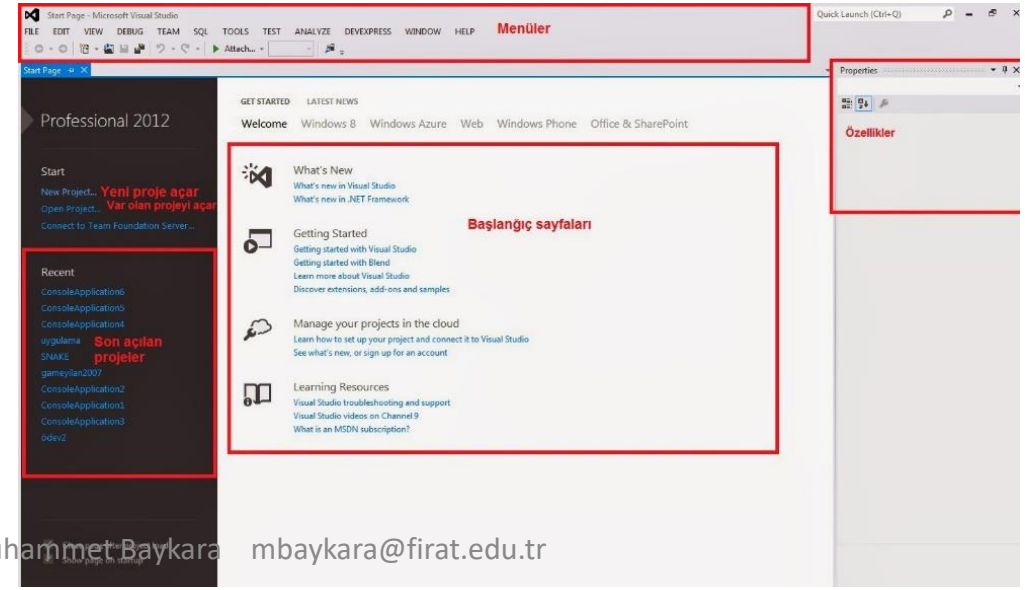
Muhammet Baykara mbaykara@firat.edu.tr

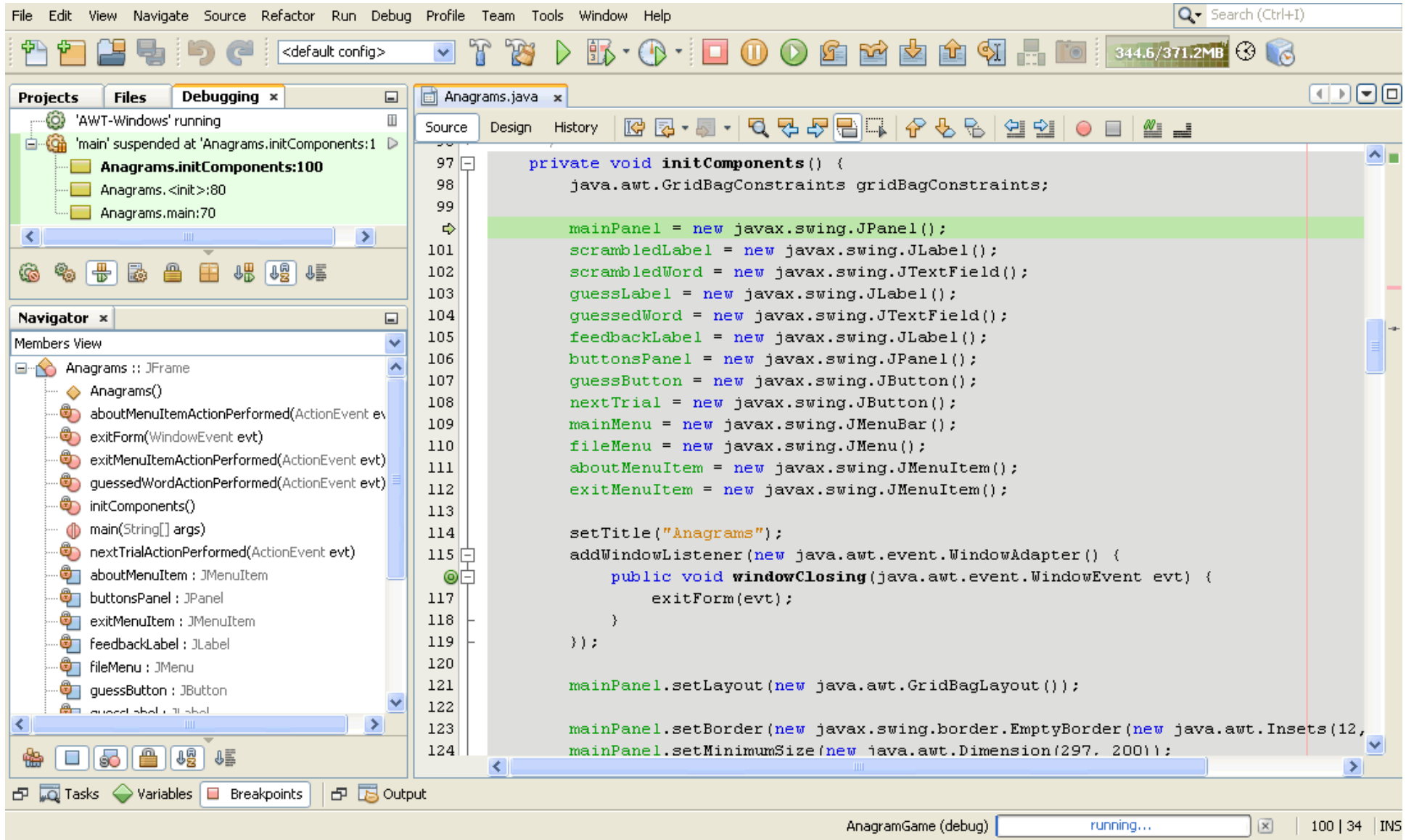


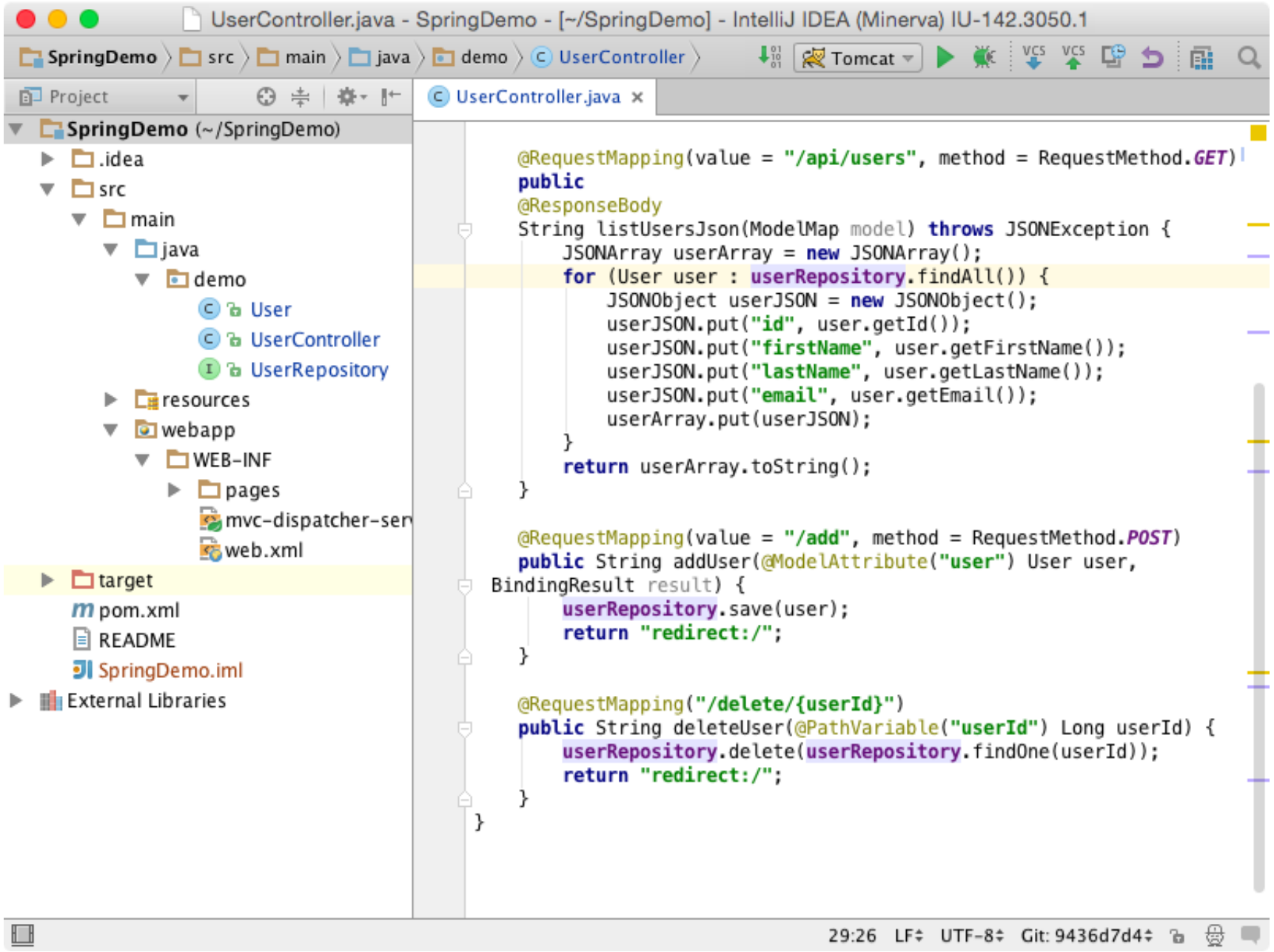
Muhammet Baykara mbaykara@firat.edu.tr

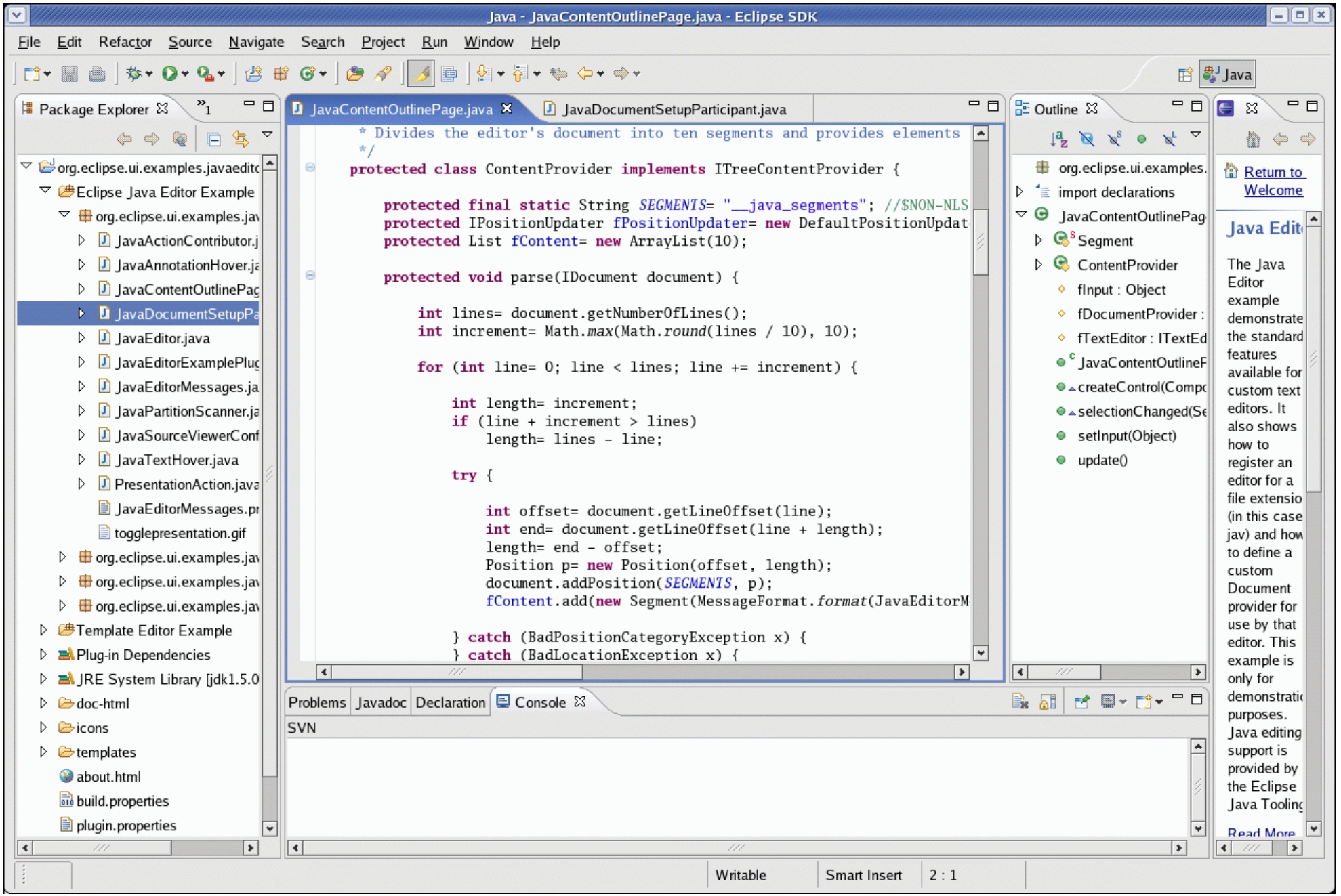
Programlama Ortamı

- Programlama ortamı, programlama dili ile birlikte birçok bileşenden oluşur.
- Bu bileşenler, sembolik olarak tasarlamış programın bilgisayar donanımı tarafından istenilen işlevleri yerine getirmesi için gereken tüm unsurlardır.
- Programlama ortamının temel unsurları Editör (Editor), Derleyici (Compiler), Kütüphane (Library), Bağlayıcı, Hata ayıklayıcı (Debugger) ve yorumlayıcı (Interpreter) dır.




File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
344.6/371.2MB
Projects Files Debugging
'AWT-Windows' running
'main' suspended at 'Anagrams.initComponents:1
Anagrams.initComponents:100
Anagrams.<init>:80
Anagrams.main:70
Navigator
Members View
Anagrams :: JFrame
Anagrams()
aboutMenuItemActionPerformed(ActionEvent ev
exitForm(WindowEvent evt)
exitMenuItemActionPerformed(ActionEvent evt)
guessedWordActionPerformed(ActionEvent evt)
initComponents()
main(String[] args)
nextTrialActionPerformed(ActionEvent evt)
aboutMenuItem : JMenuItem
buttonsPanel : JPanel
exitMenuItem : JMenuItem
feedbackLabel : JLabel
fileMenu : JMenu
guessButton : JButton
Anagrams.java
Source Design History
97 private void initComponents() {
98 java.awt.GridBagConstraints gridBagConstraints;
99
mainPanel = new javax.swing.JPanel();
101 scrambledLabel = new javax.swing.JLabel();
102 scrambledWord = new javax.swing.JTextField();
103 guessLabel = new javax.swing.JLabel();
104 guessedWord = new javax.swing.JTextField();
105 feedbackLabel = new javax.swing.JLabel();
106 buttonsPanel = new javax.swing.JPanel();
107 guessButton = new javax.swing.JButton();
108 nextTrial = new javax.swing.JButton();
109 mainMenu = new javax.swing.JMenuBar();
110 fileMenu = new javax.swing.JMenu();
111 aboutMenuItem = new javax.swing.JMenuItem();
112 exitMenuItem = new javax.swing.JMenuItem();
113
114 setTitle("Anagrams");
115 addWindowListener(new java.awt.event.WindowAdapter() {
public void windowClosing(java.awt.event.WindowEvent evt) {
117 exitForm(evt);
118 }
119 });
120
121 mainPanel.setLayout(new java.awt.GridBagLayout());
122
123 mainPanel.setBorder(new javax.swing.border.EmptyBorder(new java.awt.Insets(12,
124 mainPanel.setMinimumSize(new java.awt.Dimension(297, 200));
Tasks Variables Breakpoints Output
AnagramGame (debug) running... 100 | 34 INS

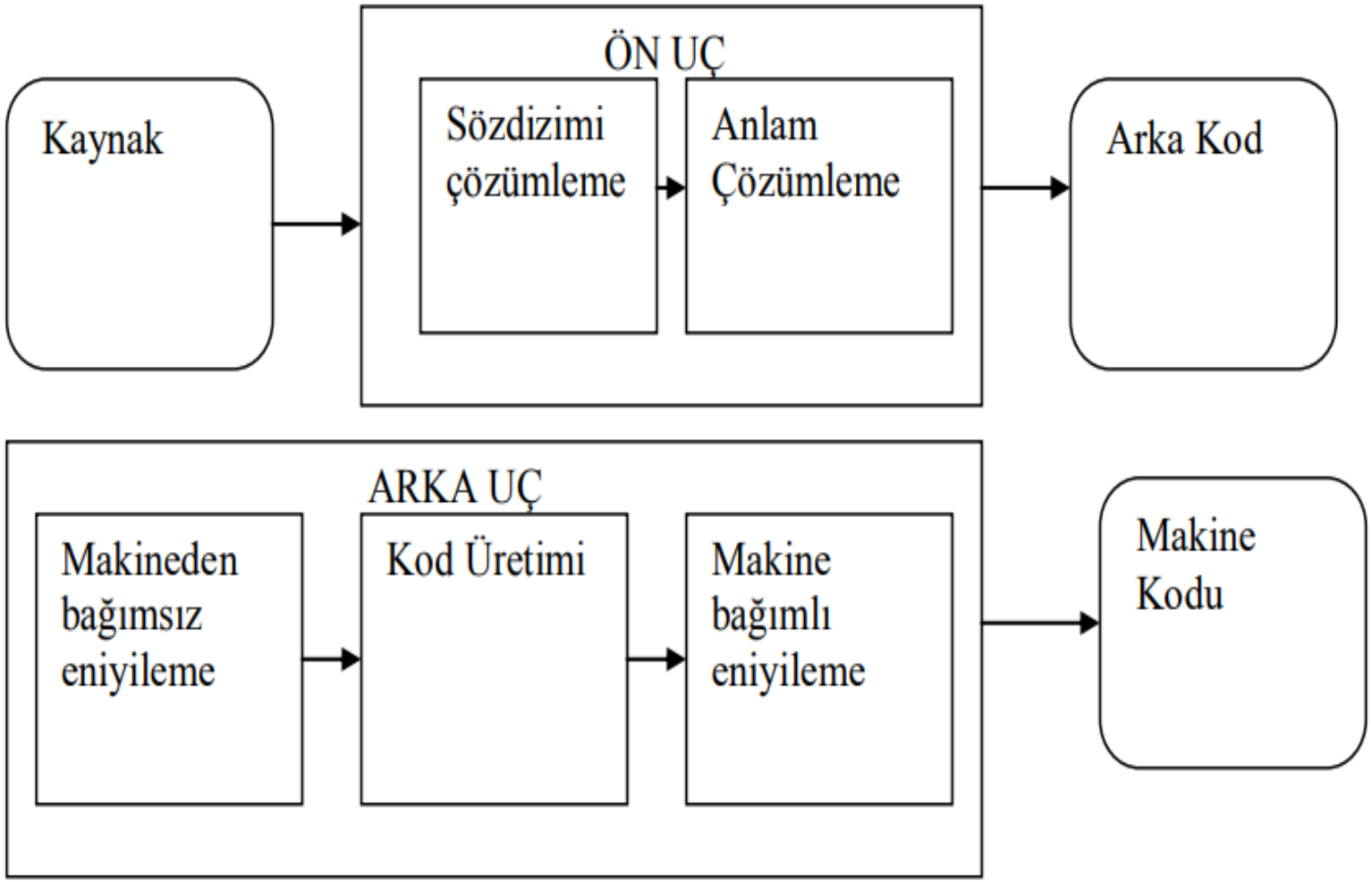

UserController.java - SpringDemo - [~/SpringDemo] - IntelliJ IDEA (Minerva) IU-142.3050.1
SpringDemo
src
main
java
demo
UserController
Tomcat
Project
UserController.java
SpringDemo (~/SpringDemo)
.idea
src
main
java
demo
User
UserController
UserRepository
resources
webapp
WEB-INF
pages
mvc-dispatcher-ser
web.xml
target
pom.xml
README
SpringDemo.iml
External Libraries
@RequestMapping(value = "/api/users", method = RequestMethod.GET)
public
@ResponseBody
String listUsersJson(ModelMap model) throws JSONException {
JSONArray userArray = new JSONArray();
for (User user : userRepository.findAll()) {
JSONObject userJSON = new JSONObject();
userJSON.put("id", user.getId());
userJSON.put("firstName", user.getFirstName());
userJSON.put("lastName", user.getLastName());
userJSON.put("email", user.getEmail());
userArray.put(userJSON);
}
return userArray.toString();
}
@RequestMapping(value = "/add", method = RequestMethod.POST)
public String addUser(@ModelAttribute("user") User user,
BindingResult result) {
userRepository.save(user);
return "redirect:/";
}
@RequestMapping("/delete/{userId}")
public String deleteUser(@PathVariable("userId") Long userId) {
userRepository.delete(userRepository.findOne(userId));
return "redirect:/";
}
}
29:26
LF
UTF-8
Git: 9436d7d4

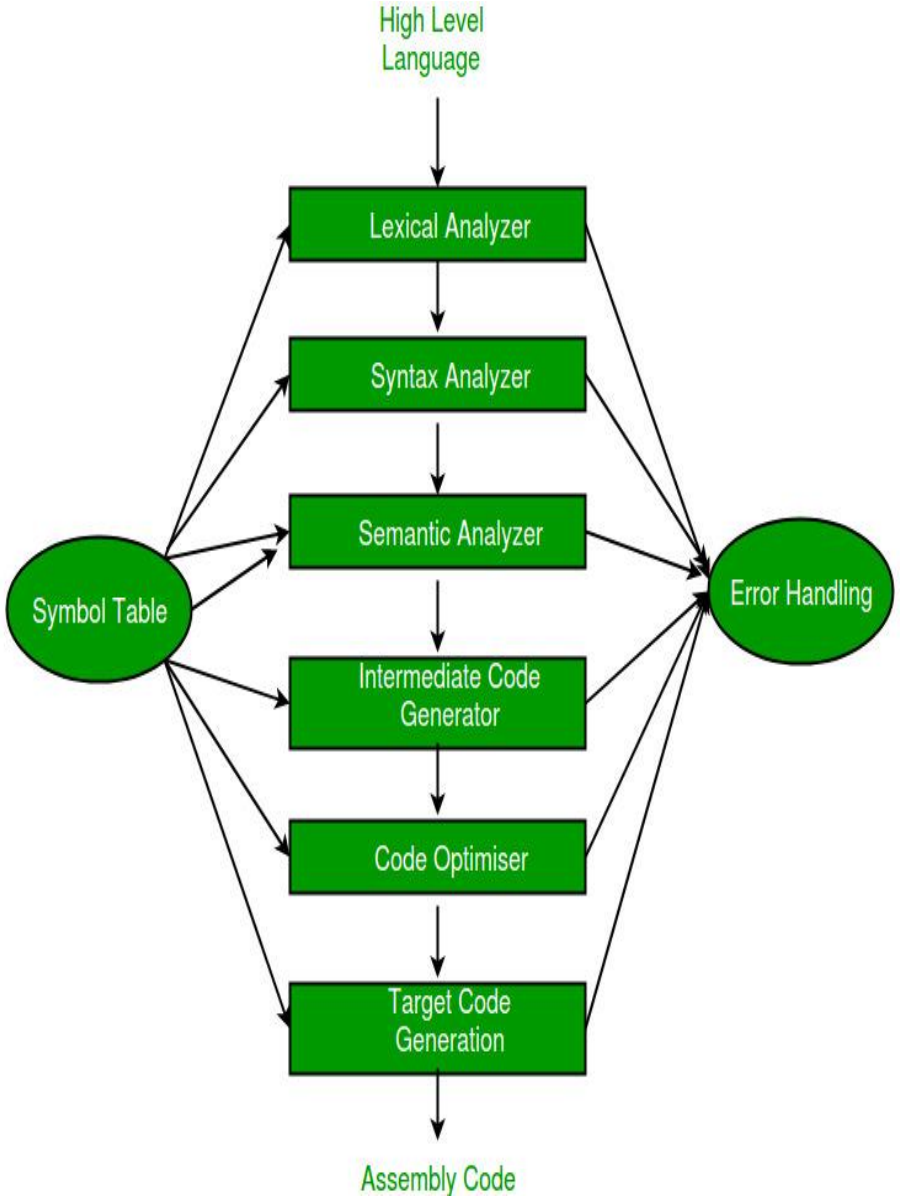

Java - JavaContentOutlinePage.java - Eclipse SDK
File Edit Refactor Source Navigate Search Project Run Window Help
Java
Package Explorer
org.eclipse.ui.examples.javaedit
Eclipse Java Editor Example
org.eclipse.ui.examples.jav
JavaActionContributor.j
JavaAnnotationHover.ja
JavaContentOutlinePag
JavaDocumentSetupPa
JavaEditor.java
JavaEditorExamplePlug
JavaEditorMessages.ja
JavaPartitionScanner.ja
JavaSourceViewerConf
JavaTextHover.java
PresentationAction.java
JavaEditorMessages.pr
togglepresentation.gif
org.eclipse.ui.examples.jav
org.eclipse.ui.examples.jav
org.eclipse.ui.examples.jav
Template Editor Example
Plug-in Dependencies
JRE System Library [jdk1.5.0
doc-html
icons
templates
about.html
build.properties
plugin.properties
JavaContentOutlinePage.java
JavaDocumentSetupParticipant.java
* Divides the editor's document into ten segments and provides elements
*/
protected class ContentProvider implements ITreeContentProvider {
protected final static String SEGMENTS= "__java_segments"; //$NON-NLS
protected IPositionUpdater fPositionUpdater= new DefaultPositionUpdat
protected List fContent= new ArrayList(10);
protected void parse(IDocument document) {
int lines= document.getNumberOfLines();
int increment= Math.max(Math.round(lines / 10), 10);
for (int line= 0; line < lines; line += increment) {
int length= increment;
if (line + increment > lines)
length= lines - line;
try {
int offset= document.getLineOffset(line);
int end= document.getLineOffset(line + length);
length= end - offset;
Position p= new Position(offset, length);
document.addPosition(SEGMENTS, p);
fContent.add(new Segment(MessageFormat.format(JavaEditorM
} catch (BadPositionCategoryException x) {
} catch (BadLocationException x) {
Outline
org.eclipse.ui.examples.
import declarations
JavaContentOutlinePag
Segment
ContentProvider
fInput : Object
fDocumentProvider :
fTextEditor : ITextEd
JavaContentOutlineF
createControl(Compc
selectionChanged(Se
setInput(Object)
update()
Return to Welcome
Java Edit
The Java Editor example demonstrate the standard features available for custom text editors. It also shows how to register an editor for a file extensio (in this case jav) and how to define a custom Document provider for use by that editor. This example is only for demonstrati purposes. Java editing support is provided by the Eclipse Java Tooling
Read More
Problems Javadoc Declaration Console
SVN
Writable Smart Insert 2 : 1

Programlama Ortamı

- **Editör (Editor):** Kaynak kodu oluşturmak ve gerektiğinde değişiklik yapmak için gerekli olan araçtır. Editörde yazılanlar seçilen dilin komutlarından oluşan metinlerdir.
- **Derleyici (Compiler):** Editörde yazılan kaynak kodu makine koduna çeviren bir programdır.
- **Kütüphane (Library):** Nesne dosyalarından oluşur.
- **Bağlayıcı:** Programın içerdiği tüm nesne dosyalarını birleştirerek tek dosya haline getirerek yürütülebilir bir dosya haline getirir.
- **Hata ayıklayıcı (Debugger):** Programcının hataları ayıklayabilmesi için programın adım adım yürütülmesini sağlar.
- **Yorumlayıcı (Interpreter):** Programın kaynak kodunu doğrudan satır satır yürüten bir programdır.

Bir Derleyicinin Genel Yapısı





Programlama Dillerinin Elemanları

- **Sözdizimi/Yazım (Syntax):** Temel olarak bir dilde (language) tanımlı olan öğelerin (kelime, işlem, sembol yada değerlerin) anlamlı bir dizilim oluşturmasıyla ilgilenen bilimdir.

- **Anlambilim (Semantics):** Bir programlama dilindeki bir ifadenin ne anlama geldiğidir.

- **Veri (Data):** Bir programda farklı veri tipleriyle işlem yapmamız gerekebilir. Örneğin, tamsayılar, kesirli sayılar, karakterler (harfler ve klavyedeki diğer simgeler), metinler (string), mantıksal (boolean) değerler (doğru=true, yanlış=false) ilk aklımıza gelen farklı veri tipleridir

Programlama Dillerinin Elemanları

- **Atama Deyimi (Assignment Statement):** Şeklindeki satırlar birer atama deyimini (assignment statement) göstermektedir. Bu atama deyimleri ile x değişkeninin bellekteki adresinde 5 değeri ve y değişkeninin bellekteki adresinde ise 6 değeri görünecektir.

```
1 | x=5;
2 |
3 | y=6;
```

- **Tür Kontrolü:** Programlama dillerinde yapılan her işlem öncesi hataları önlemek için verilerin tip kontrolü yapılır. Bu tür kontrollere ”Checking” adı verilir. Checking işlemleri compile-time (derleme esnasında) veya run-time (çalışma esnasında) olarak yapılır. İşte bu farklılık Statik Tipli ve Dinamik Tipli dillerin temel ayrım noktalarından biridir.

Programlama Dillerinin Elemanları

- **Kontrol Deyimleri:** Program içerisinde bir işin olumlu veya olumsuz sonucuna göre yapılması gereken işlemler olabilir. Bu aşamada devreye kontrol deyimleri girer. Örnek kontrol deyimleri; If Else Deyimi, Switch-Case Deyimi.

- **Alt Programlar:** Bir programda aynı tür hesaplama işlemi programın farklı yer(ler)inde birden fazla kullanılabilir. Aynı işlem adımlarını bir çok kez tekrarlamak, programdaki deyim sayısını arttıracağından hem programın yavaş işlemesini, özellikle programı(n çalışması açısından) izlemeyi güçleştirir. Bu tür tekrarlanan program parçaları, ALT PROGRAM adı altında (ana programdan) ayrı bir program olarak yazılabilir.

```
public void denemeFonksiyonu(){
  if( s.toString().toLowerCase().indexOf("plane")>0){
    System.out.println("yer tıklandı");
  }
  else if(tg!=null){
    for(int i = 0 ;i<10;i++){
      if(tg.toString().equals(objectList[i])){
        tf.setText("");tf.setText(tf.getText()+"\n"+"Tank "+i+" tıklandı");
        tf.setText(tf.getText()+"\n"+"Tankın bölgesi :"+ tankRegion[i]);
        clickedTank = i;
        tf.setText(tf.getText()+"\n"+"Tank Owner : " + tankOwner[i] );

      }
    }
        if(tankOwner[clickedTank]!=activeplayer){
          tf.setText("");tf.setText(tf.getText()+"\n"+"Düşman tankı tıklanamaz");
          clickedTank=-1;
          return;
        }
        if(tankAction[clickedTank]<1){
          tf.setText("\n"+"Tank bu sene başka yere gidemez!"+tf.getText());
          clickedTank=-1;
          return;
        }
  }
 }
public static void main(String args[]{
  denemeFonksiyonu();
}
```

F
G
E
B
A
C
D
H

Muhammet Baykara mbaykara@firat.edu.tr

Programlama Dillerinin Elemanları

- **Modüller**: programlama (modular programming) tekniğinde, birbirleriyle ilgili olduğu düşünülen ya da aynı çağrışım kümesi içinde olduğu düşünülen fonksiyonlar bir araya toplanarak genellikle farklı bir dosya içinde tutulurlar.

Kaynaklar

http://content.lms.sabis.sakarya.edu.tr/Uploads/50101/39475/bmg_programlamadilleri.pdf
http://www.farukkirmizi.com/programlama-ve-programlama-ortami.html
https://algoritmaveprogramlama.wordpress.com/2013/09/21/programlama-terimleri-ve-programlama-ortami/
https://www.baskent.edu.tr/~tkaracay/etudio/ders/prg/java/ch05/dataTypes.htm
https://mhmtyylc.com/c-sharp-kontrol-deyimleri
https://acikders.ankara.edu.tr/pluginfile.php/20859/mod_resource/content/1/fzm205_8.pdf