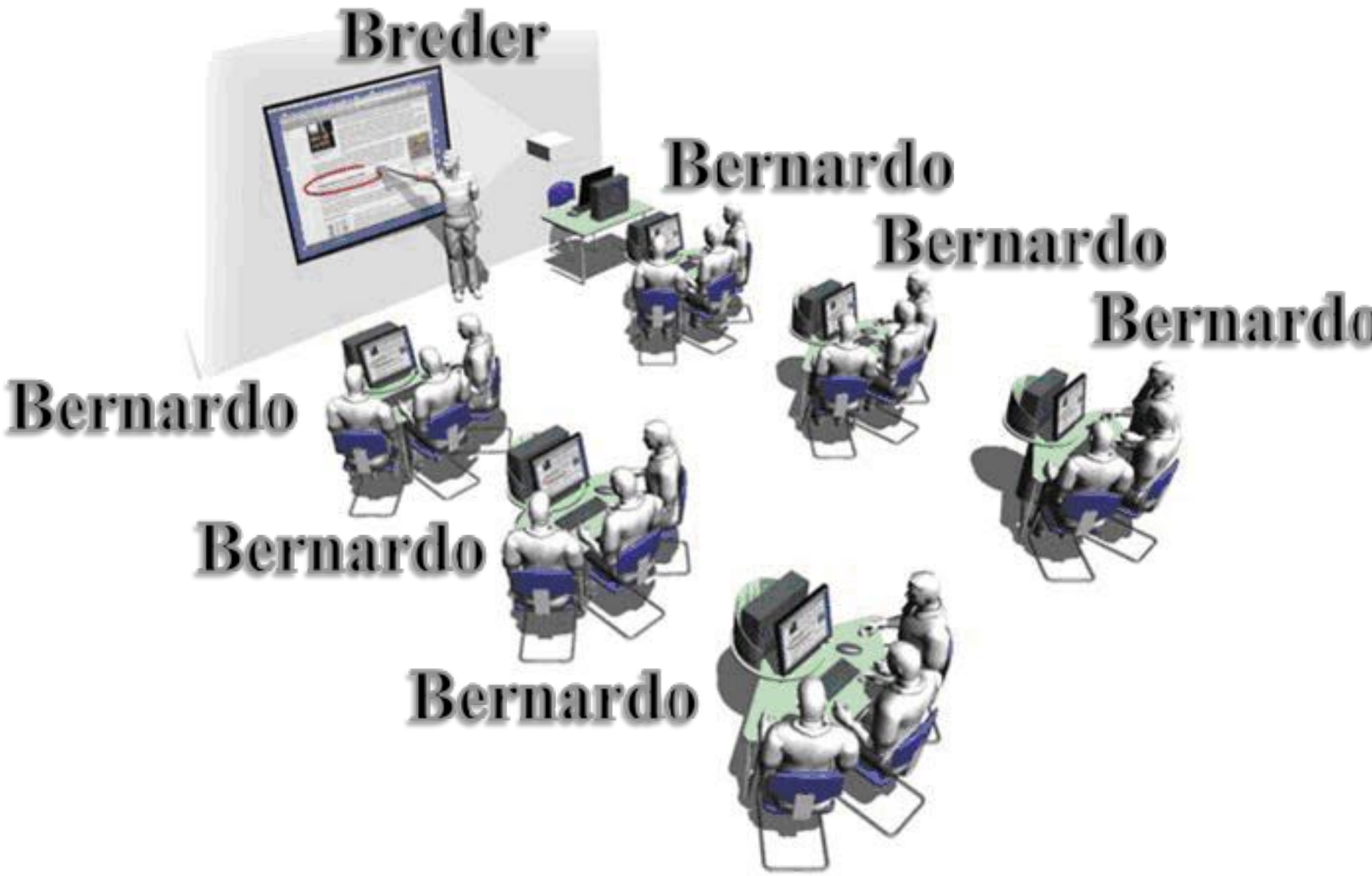
Breder Language

Bernardo Breder

03/09/2010

Teatro


Breder
Bernardo
Bernardo
Bernardo
Bernardo
Bernardo
Bernardo
Bernardo

History of the Language

Xerox Company (2007 – 2008)

- Na empresa Xerox Company, desenvolvi o primeiro compilador para geração de Relatório.

Inicio do Projeto (2008)

- Foi iniciado o projeto depois que saí do Xerox Company, com o objetivo de criar uma Linguagem mais complexa.

Graduação (2005 – 2010)

- O Escopo definido na Graduação foi de implementar os aspectos básicos da Linguagem.
- A Linguagem foi usada para a monografia de Graduação, com o objetivo de reforçar a sua seriedade.

Paper

- SBGames 2010 – Breder Programming Language for iPhone Game.
- Em andamento – Bind de Cuda para a Linguagem Breder, com o objetivo de atender as dificuldades do ambiente.
- Em andamento – Bind de Opengl para desenvolvimento de framework Gráfico.

Mestrado (2011)

- No Mestrado, pretando investir na área de Processamento Distribuido.

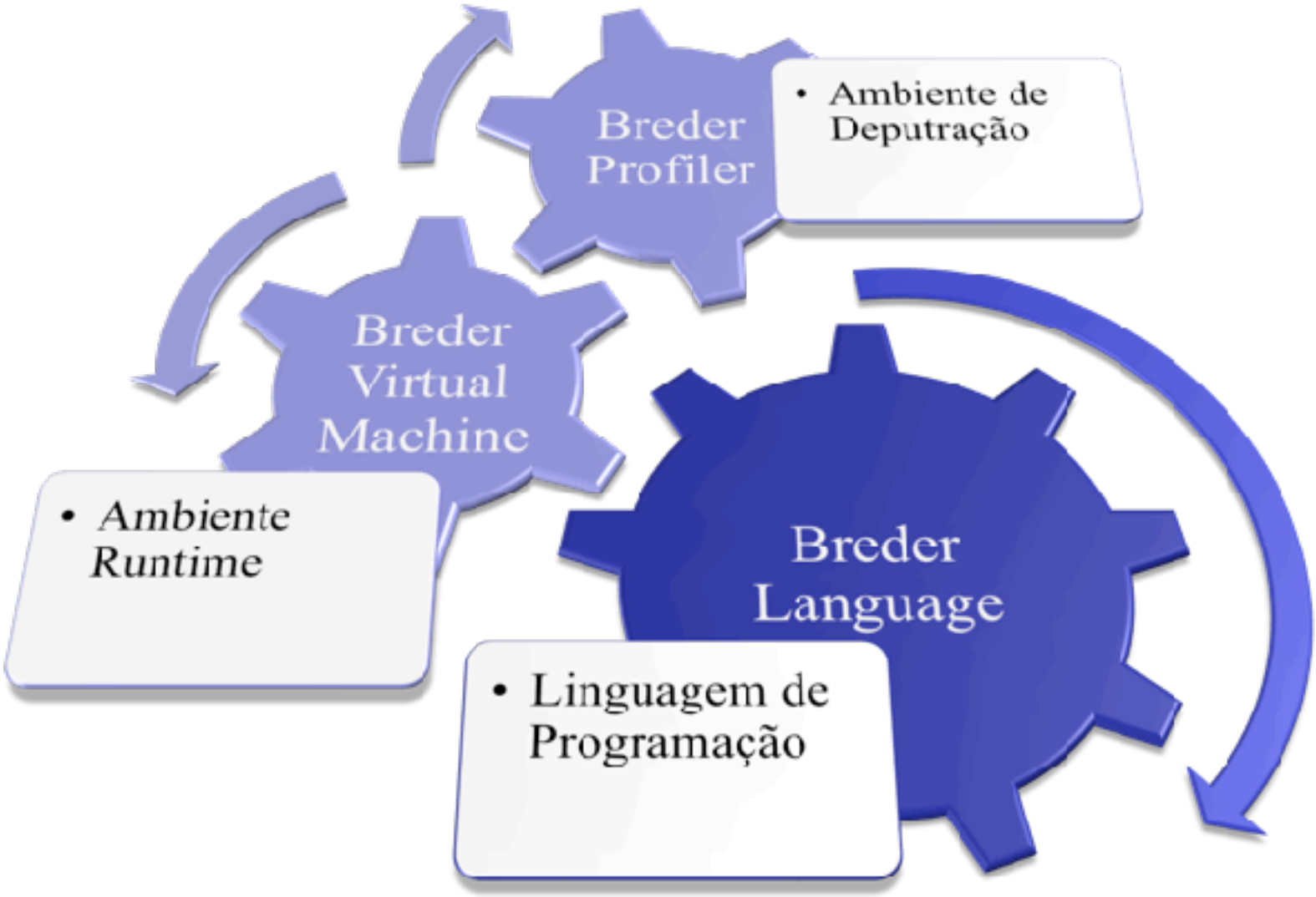
Doutorado

- Sobre analise.

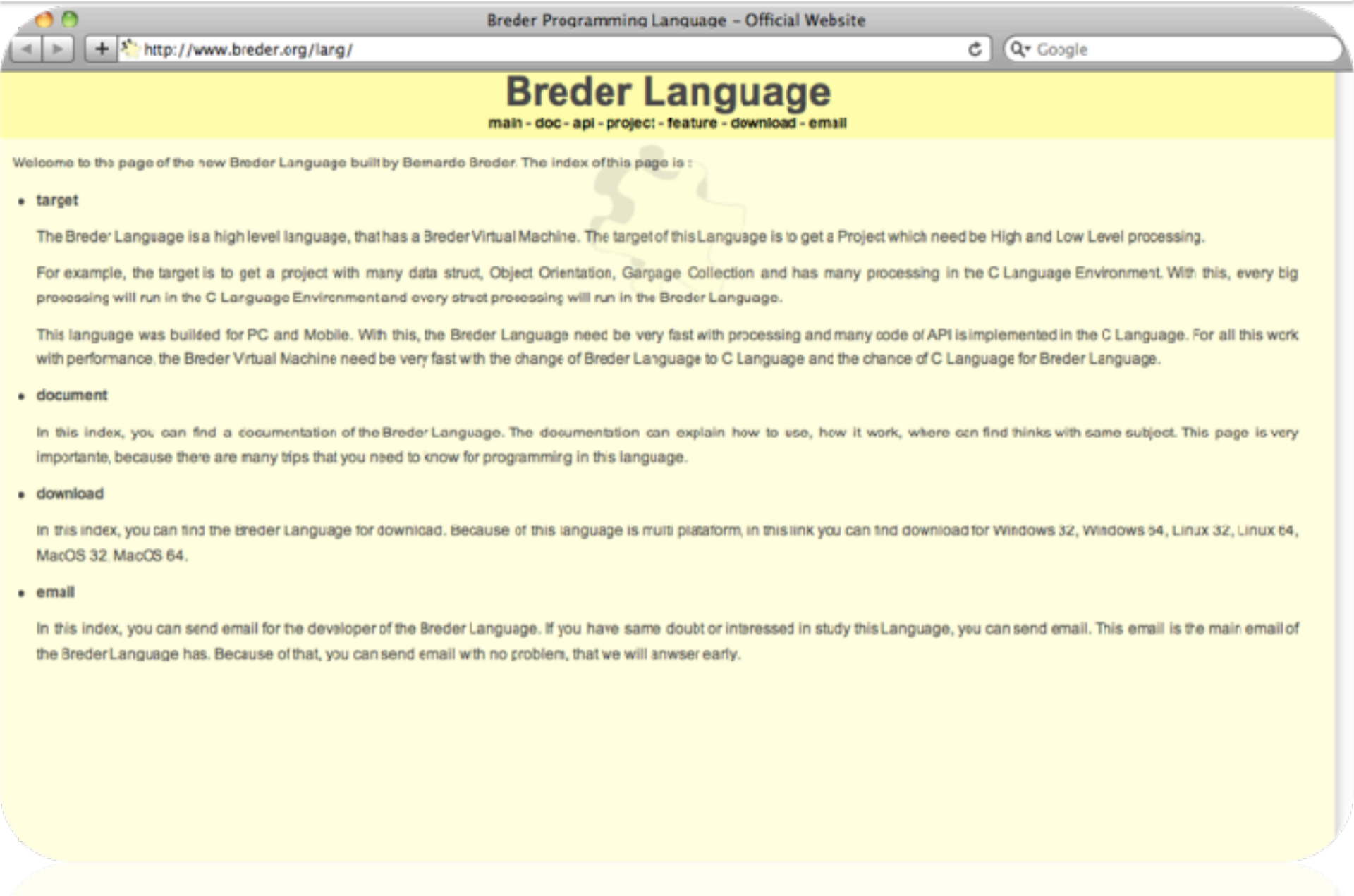
Target


Mostrar as operações em alto nível.
Mostar a importância das operações baixo nível.
Mostrar o escopo da Linguagem.
Mostrar quais são os melhores projetos para a Linguagem
Mostrar resultados.

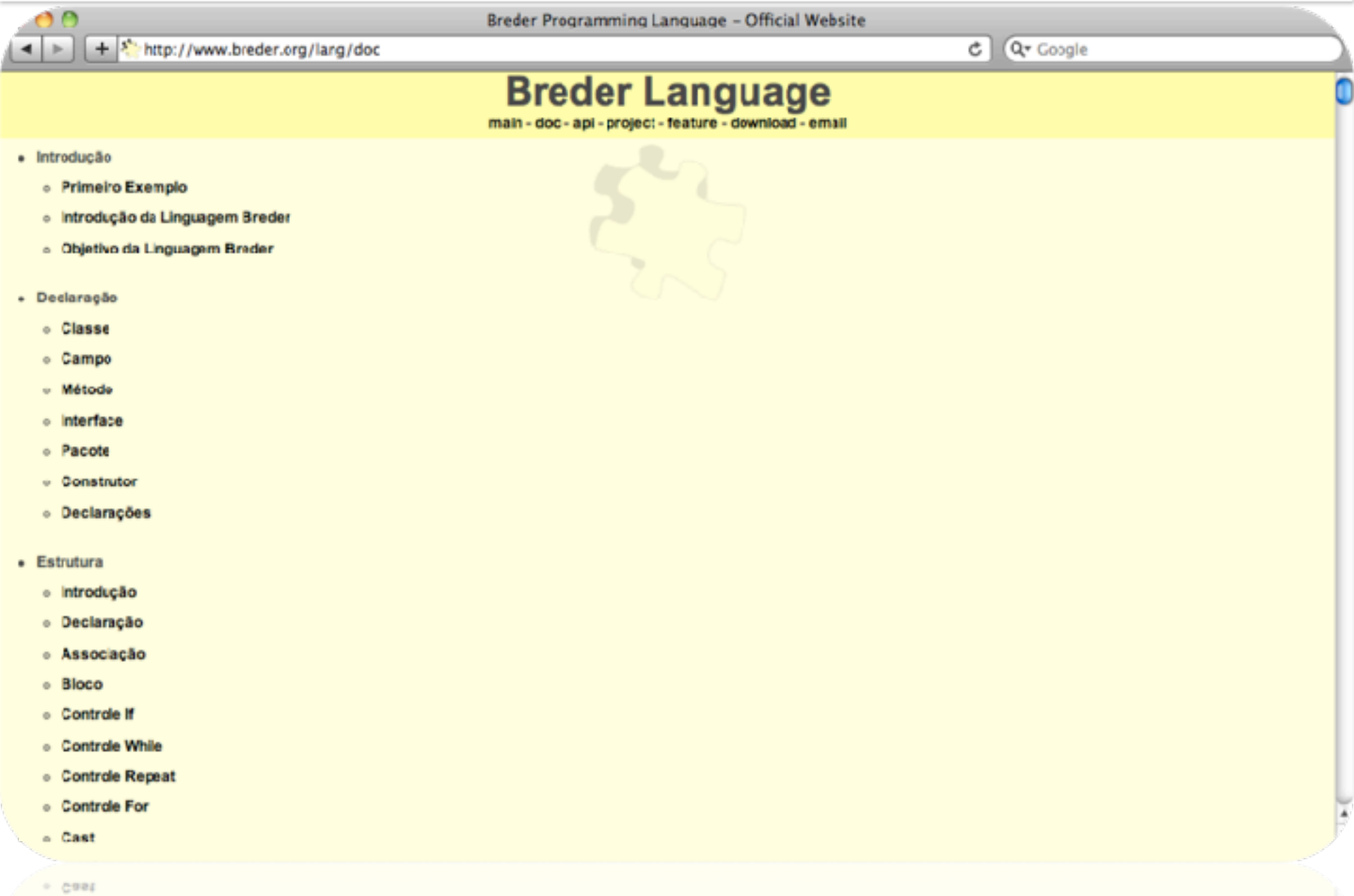
Product


Breder Profiler
• Ambiente de Deputração
Breder Virtual Machine
• Ambiente Runtime
Breder Language
• Linguagem de Programação

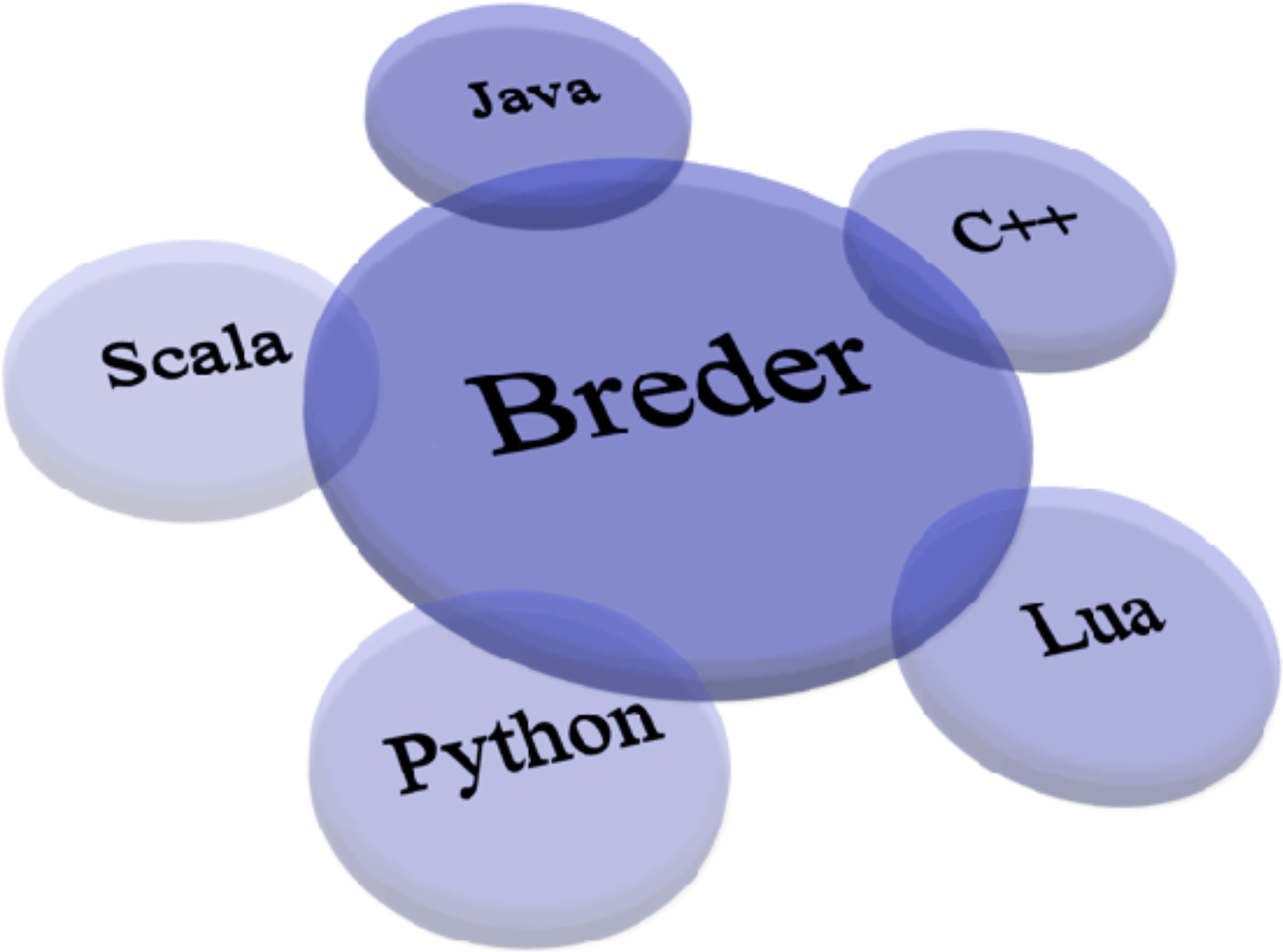
Official Web Site



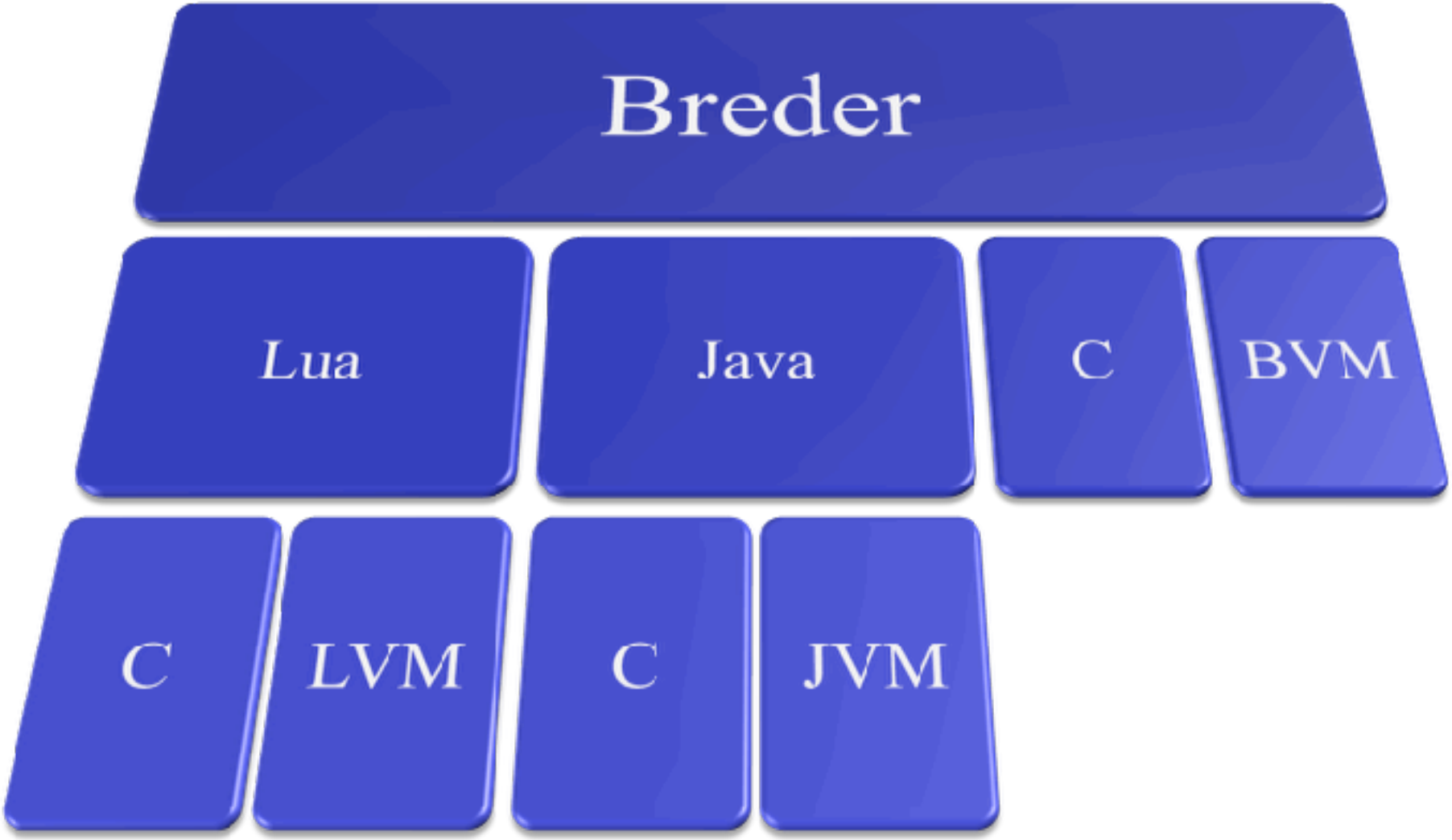
Official Web Site



High Level



High Level



High Level

```
void test () {}
```

- Executado na BVM

```
native void test () ;
```

- Executado em C Ansi

```
lua void test () ;
```

- Executado em LVM

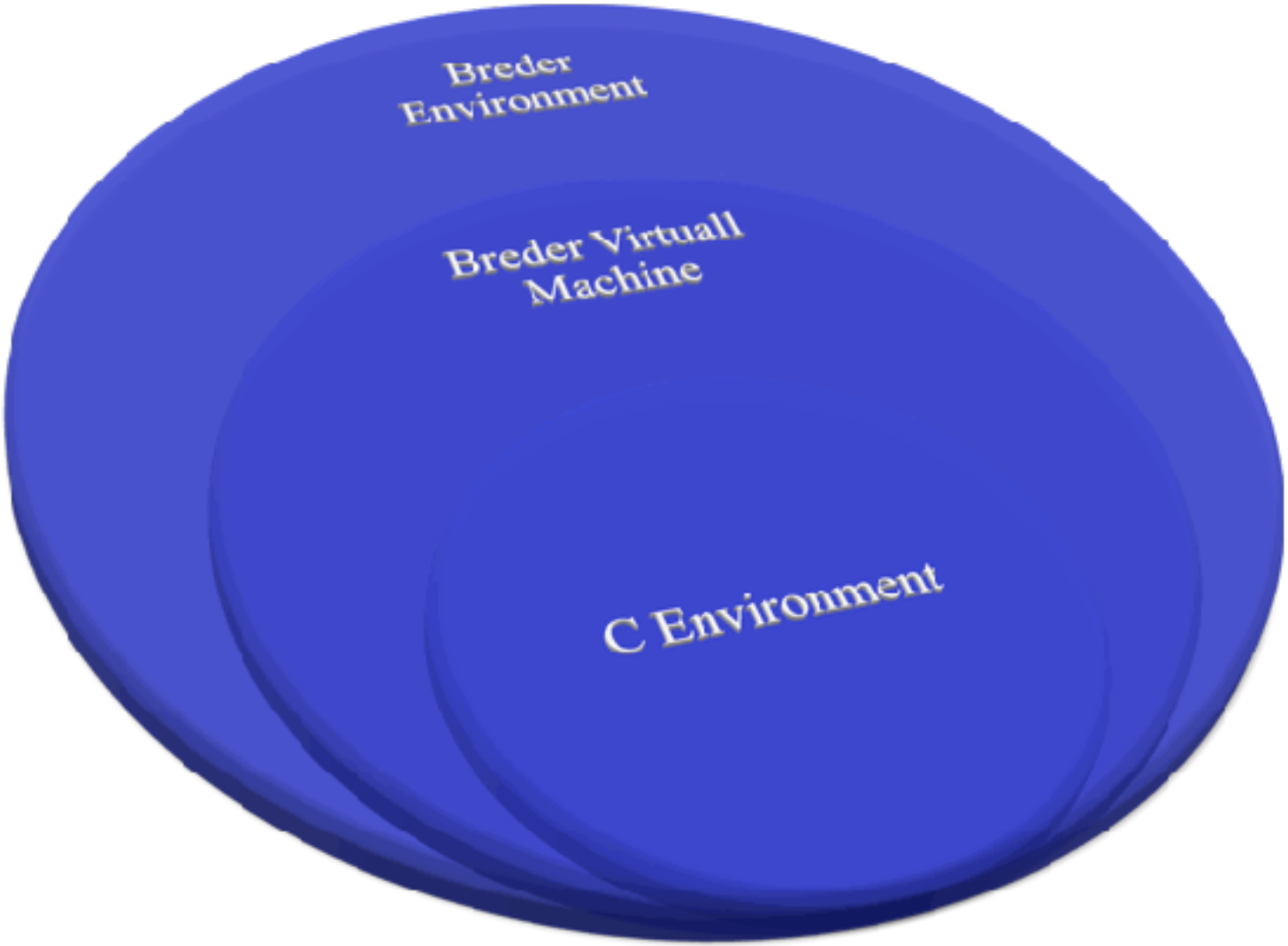
```
java void test () ;
```

- Executado em JVM

```
cuda void test () ;
```

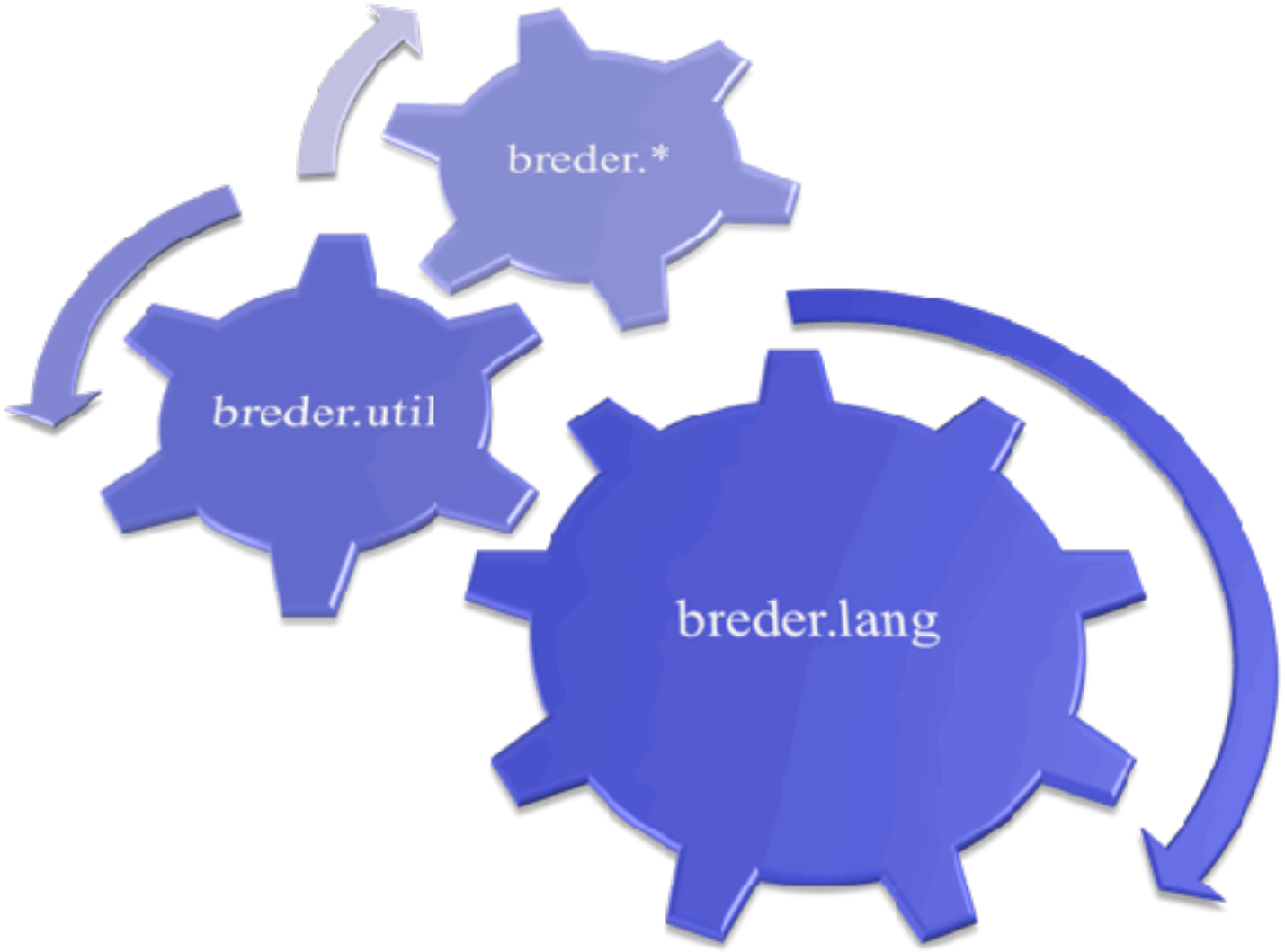
- Executado em BVM na GPU

Low Level


Breder
Environment
Breder Virtuall
Machine
C Environment

Low Level

Low Level



Low Level

breder.lang.String

- native String sum(String)
- native Boolean equal(String)

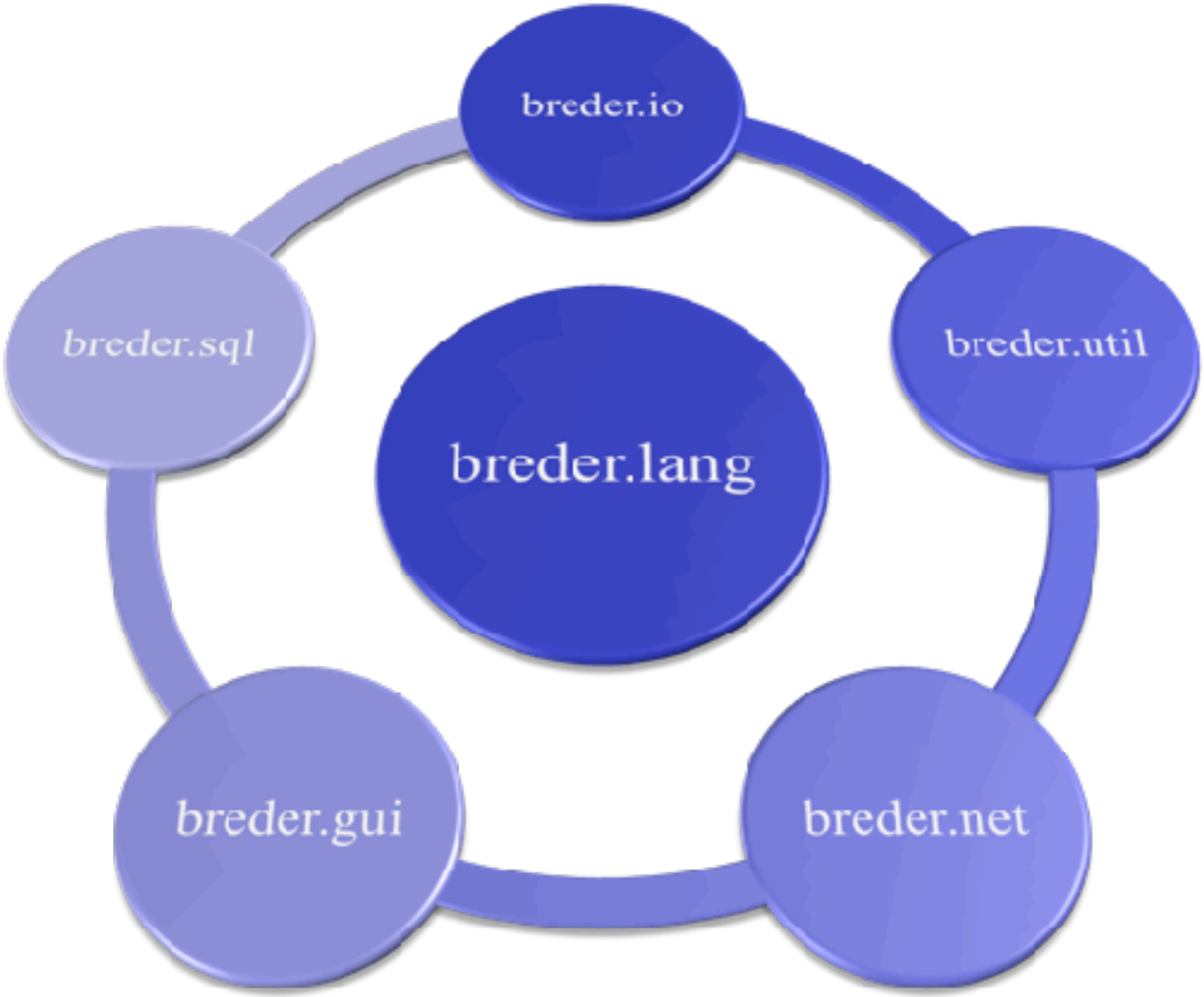
breder.util.ArrayList

- native void add(E)
- native void remove(E)

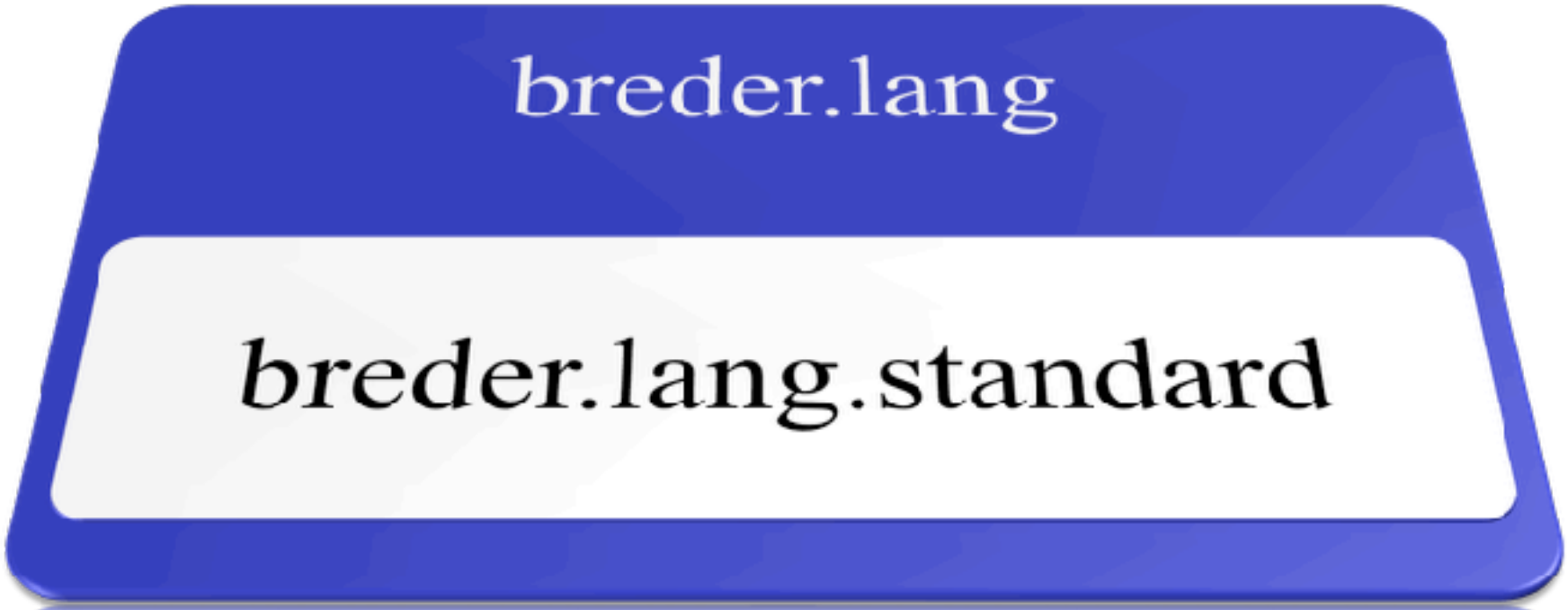
breder.io.File

- native String getName()
- native void close()

Specification

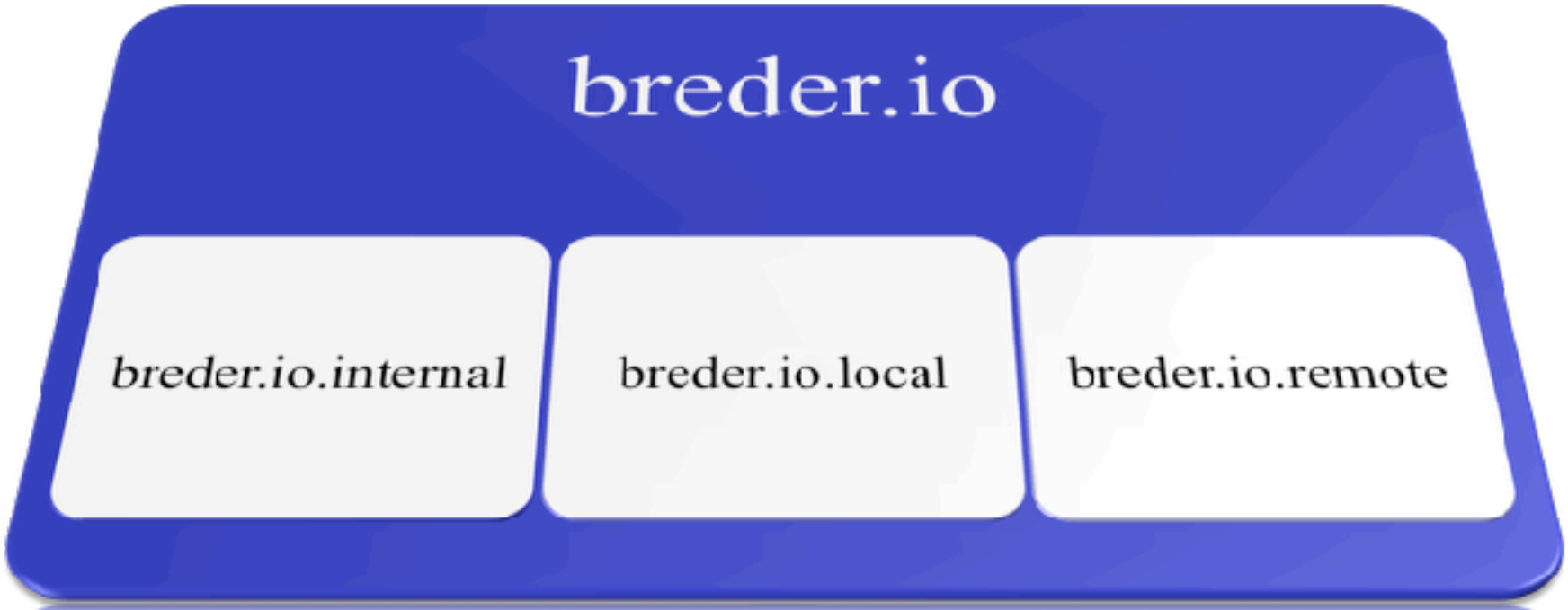


Specification



Specification

Specification



Specification

Specification

breder.lang

breder.lang.IObject

- Classe basica da Linguagem.

breder.lang.IString

- Classe que representa um array de caracteres.

breder.lang.IBoolean

- Classe que representa uma marcação de habilitado e desabilitado.

breder.lang.INumber

- Classe que representa um número qualquer, com virgula ou sem.

breder.lang.IInteger

- Classe que representa número inteiro negativo ou não.

breder.lang.INatural

- Classe que representa números natuais inteiros, de zero ao infinito.

breder.lang.IIndex

- Classe que representa número que representa um indice, de um ao infinito.

breder.util

breder.lang.IArray

- Classe que representa um array de quantidade de celula fixa.

breder.lang.IList

- Classe que representa uma lista de quantidade dinâmica, herdada da classe IArray.

breder.lang.IMap

- Classe que representa um mapa de chave e valor.

breder.lang.IStringBuilder

- Classe que responsável por construir String de forma dinâmica.

breder.io

breder.lang.IFileSystem

- Classe que representa um Sistema de Arquivo

breder.lang.IResource

- Classe que representa um recurso existente ou não.

breder.lang.IFile

- Classe que representa um arquivo existente.

breder.lang.IFolder

- Classe que representa um diretório existente.

breder.lang.IStream

- Classe que realiza transferência ou recebimento de conteúdo.

breder.lang.IInputStream

- Classe que realiza o recebimento de conteúdo.

breder.lang.IOutputStream

- Classe que realiza o transferimento de conteúdo.

breder.net

breder.lang.IServerSocket

- *Classe que representa um serviço de recebimento de requisição.*

breder.lang.IClientSocket

- *Classe que representa uma solicitação de um serviço a um servidor.*

breder.lang.ISocket

- *Classe que realiza transferência e recebimento de conteúdo.*

breder.sql

breder.lang.ISqlConnection

- Classe que representa uma conexão com um banco Sql.

breder.lang.ISqlTransaction

- Classe que representa uma Transação, existente ou não.

breder.lang.ISqlPrimitive

- Classe primitivas do Sql.

Local & Dist Processing

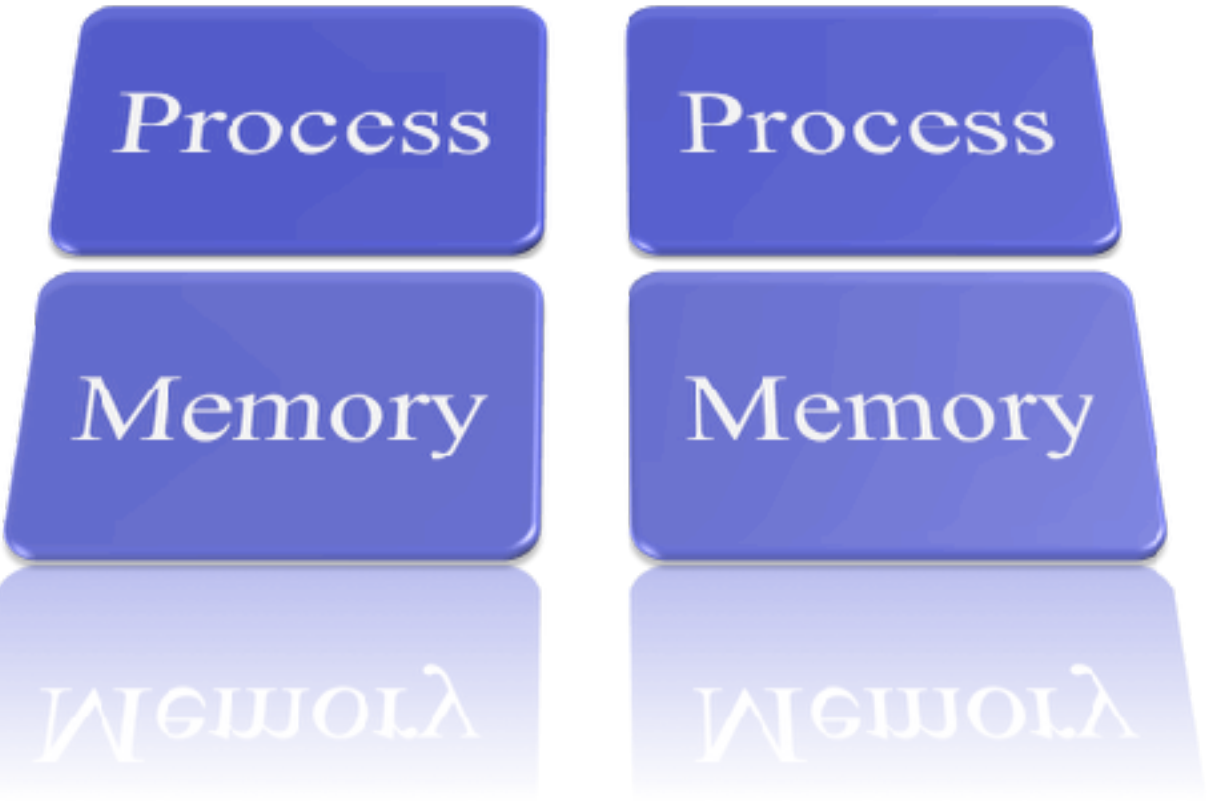
Programação Local

- *1* Processo na BVM

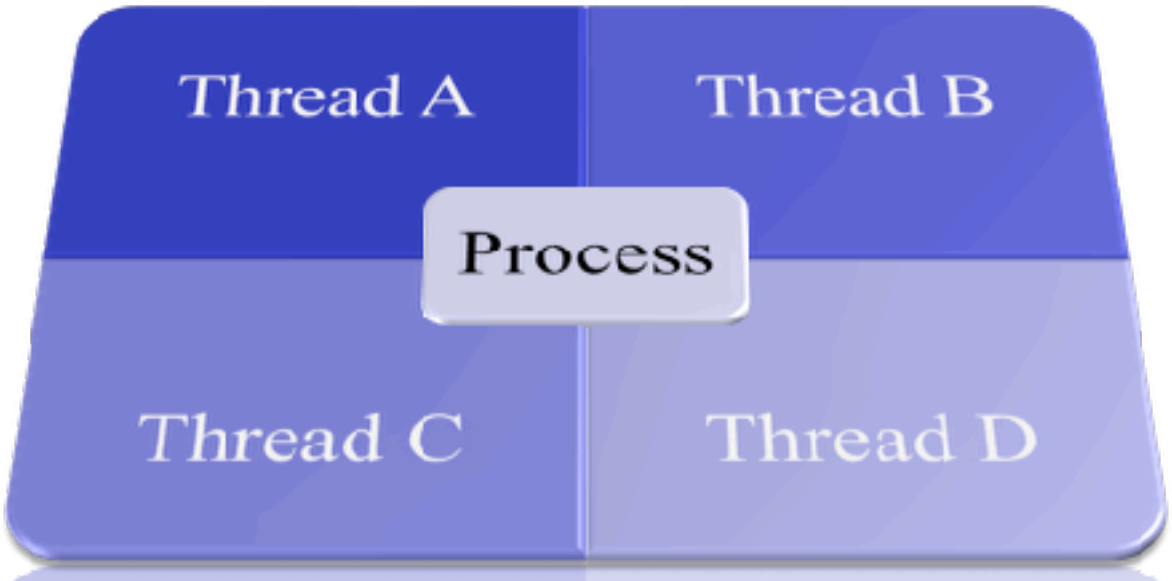
Programação Distribuida

- *N* Processos na BVM

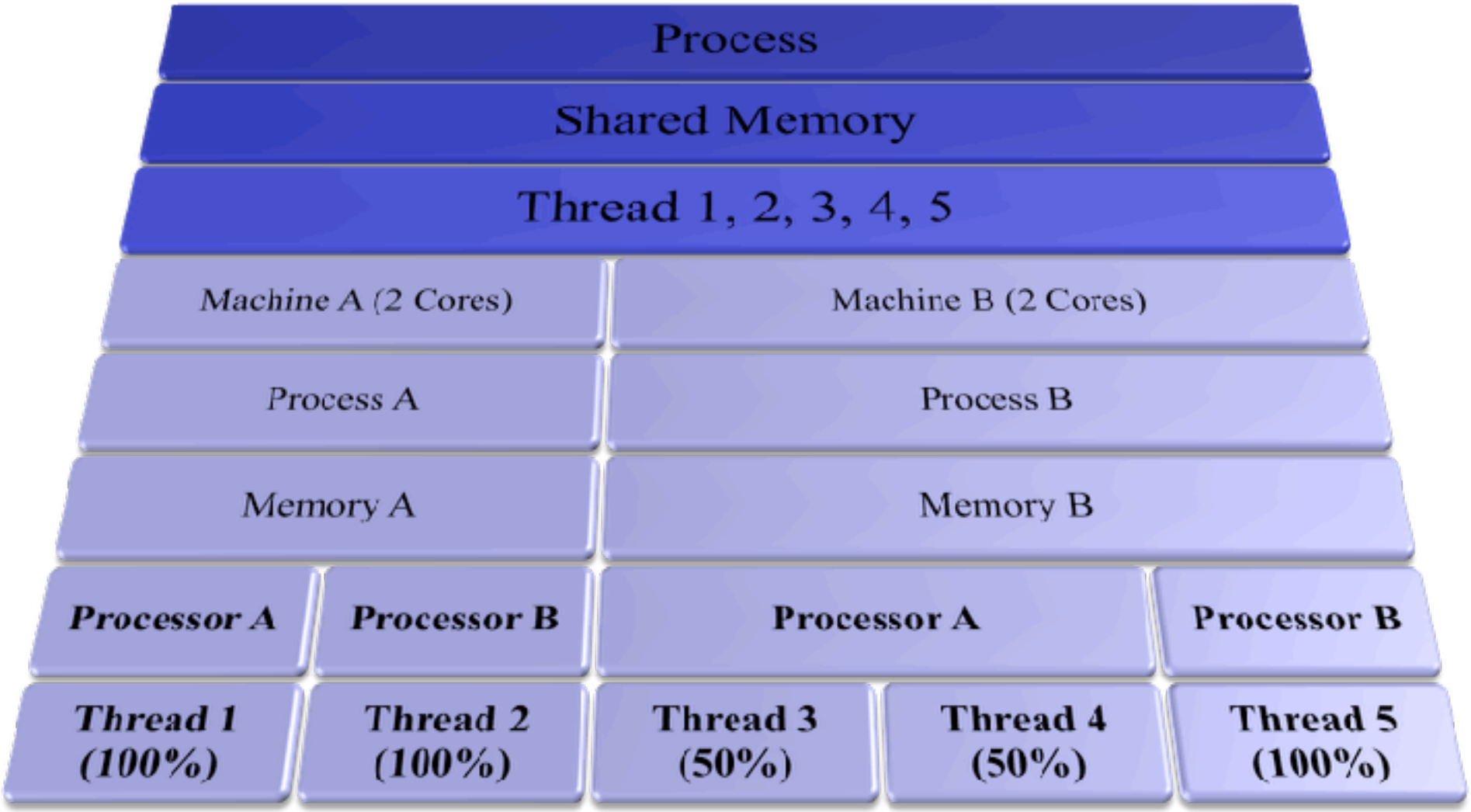
Local Processing



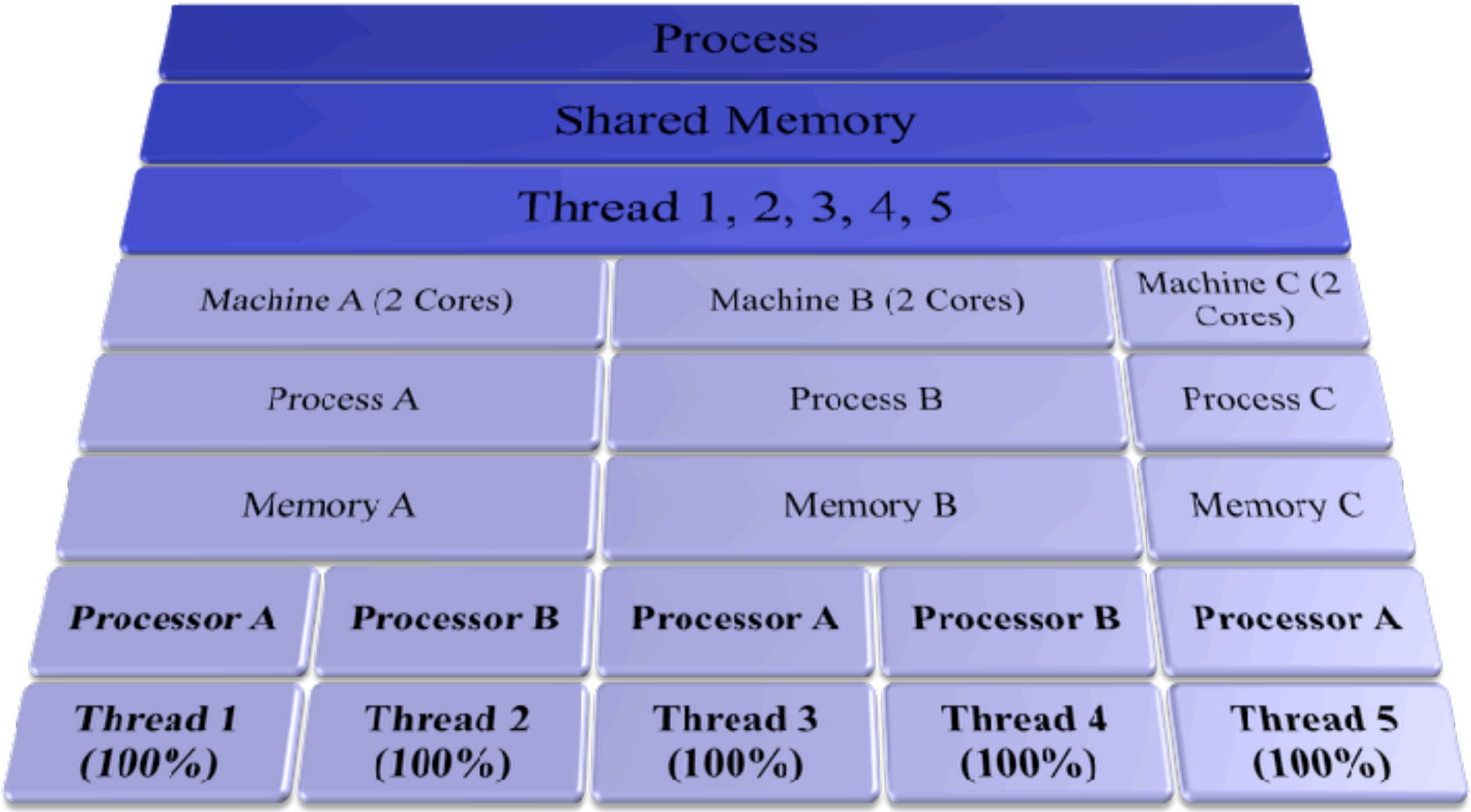
Local Processing



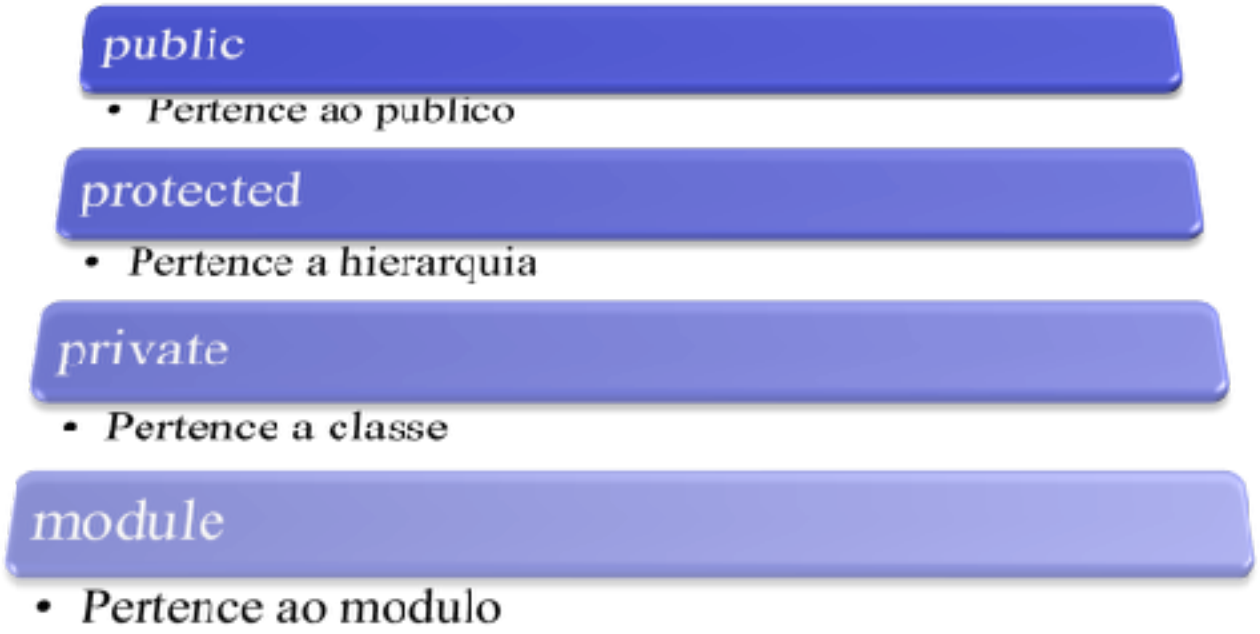
Distributed Processing



Distributed Processing

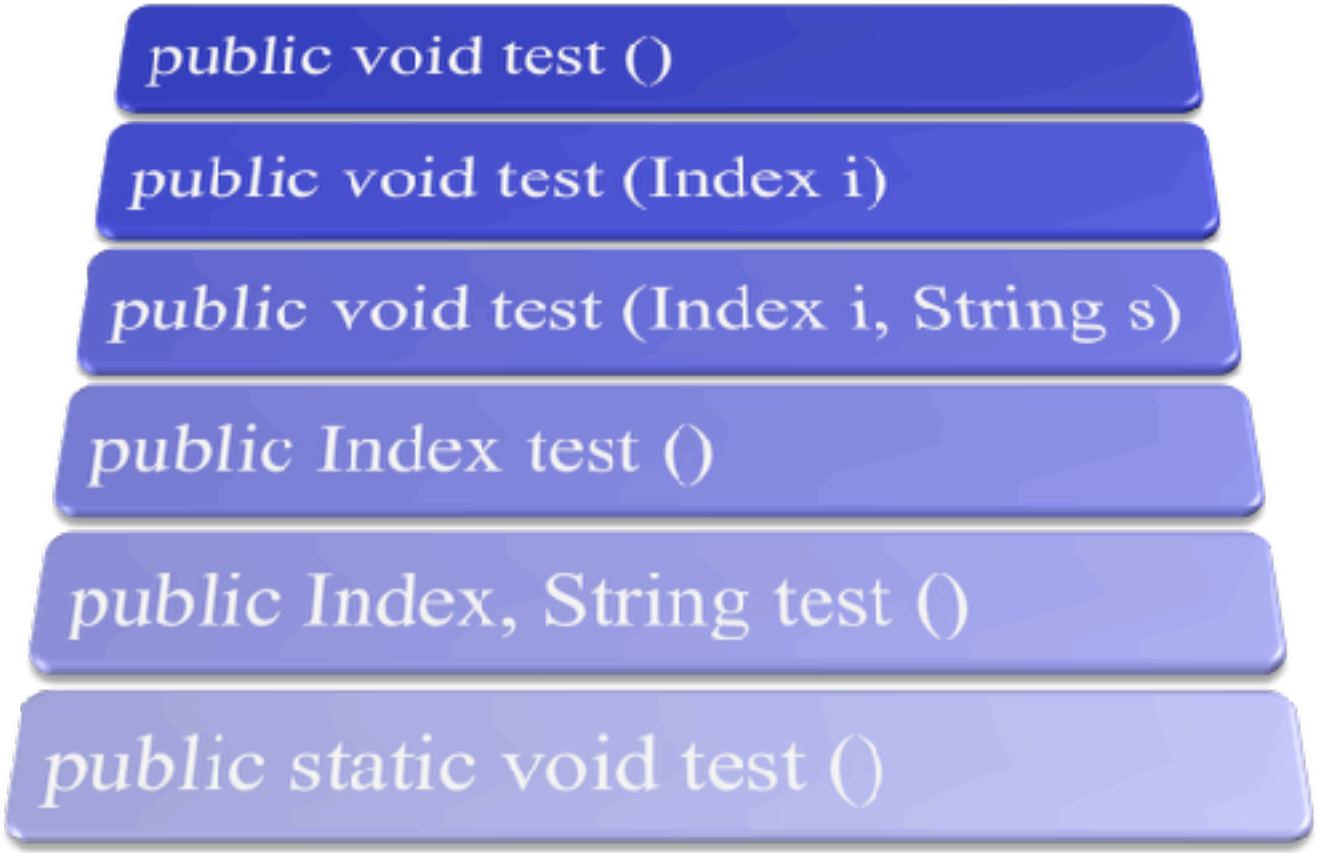


Acess Modifier

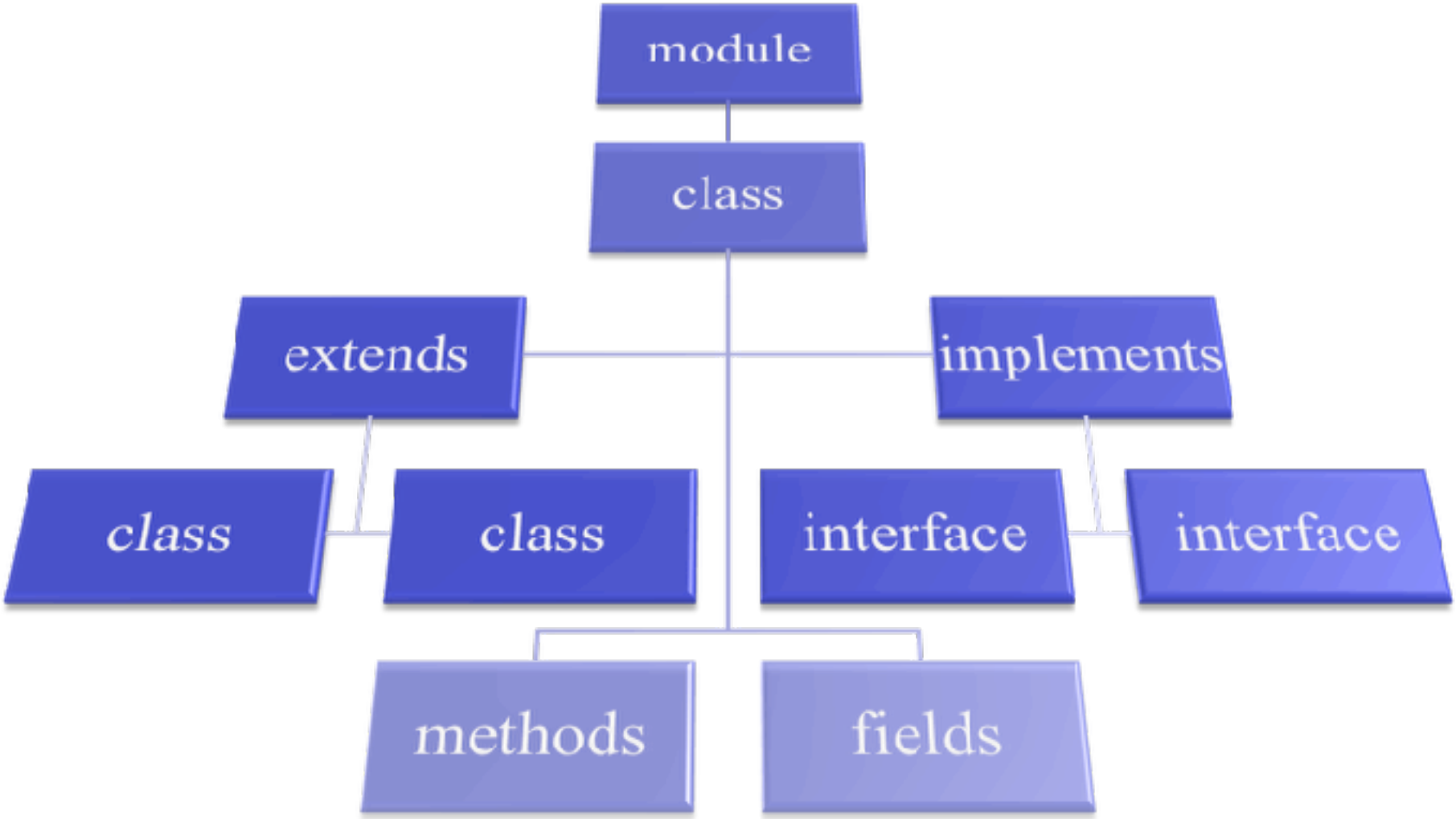


Field Declaration

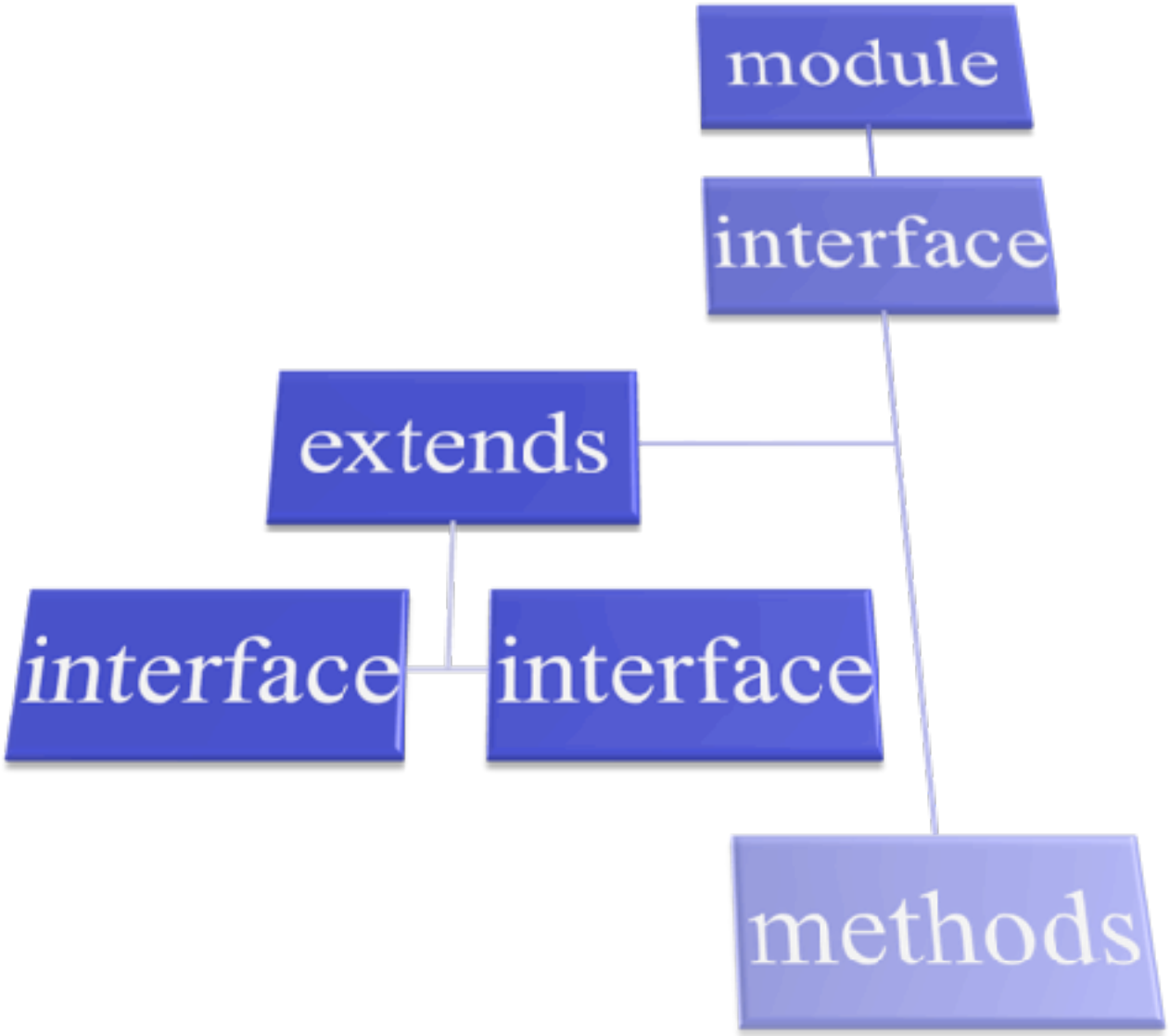
Method Declaration



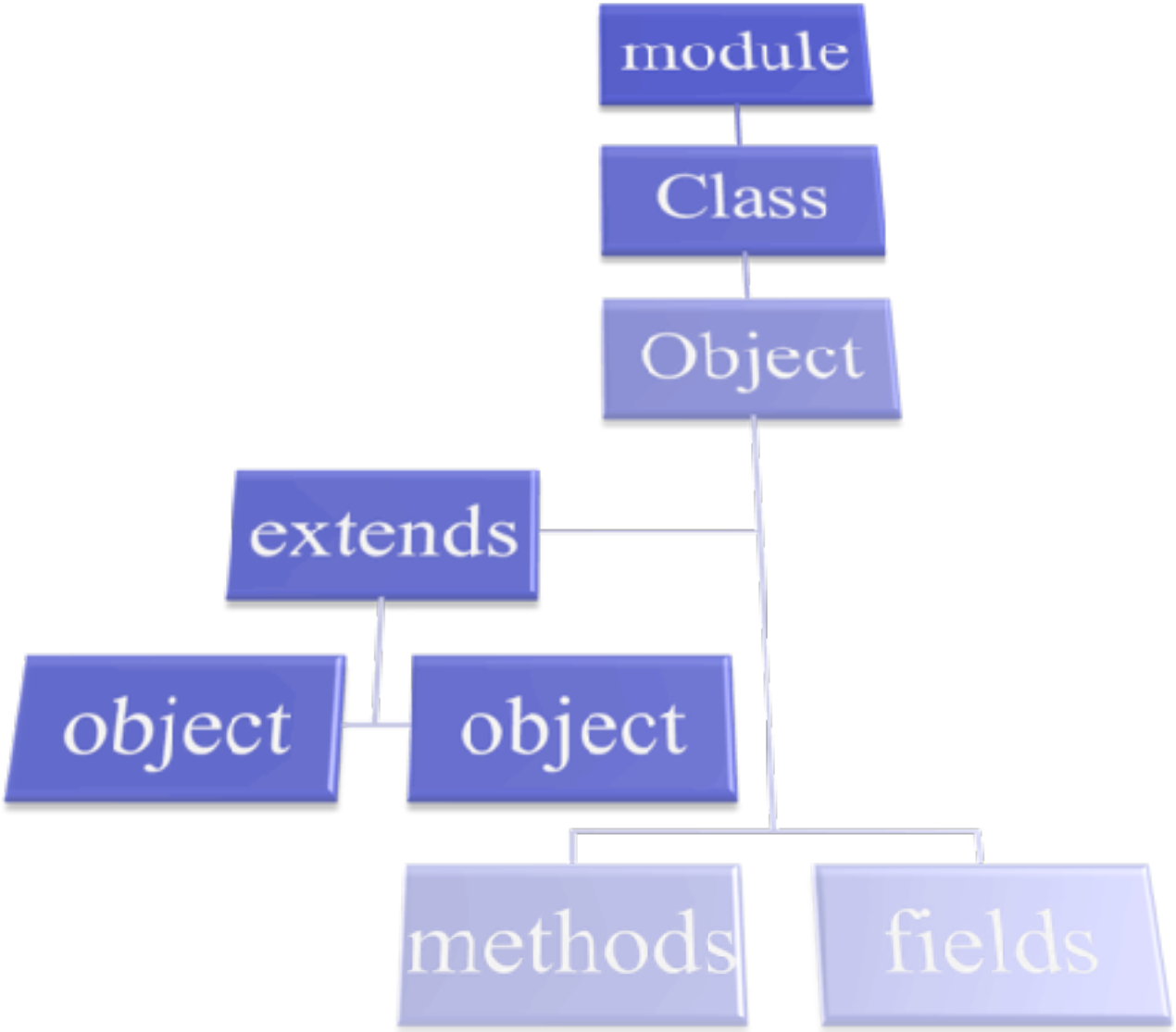
Class


module
class
extends
implements
class
class
interface
interface
methods
fields

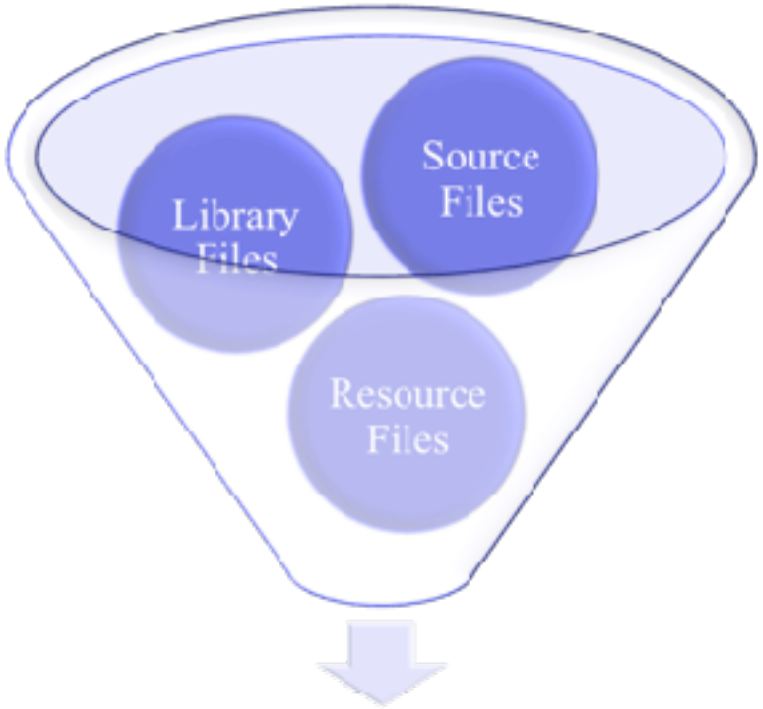
Interface



Object



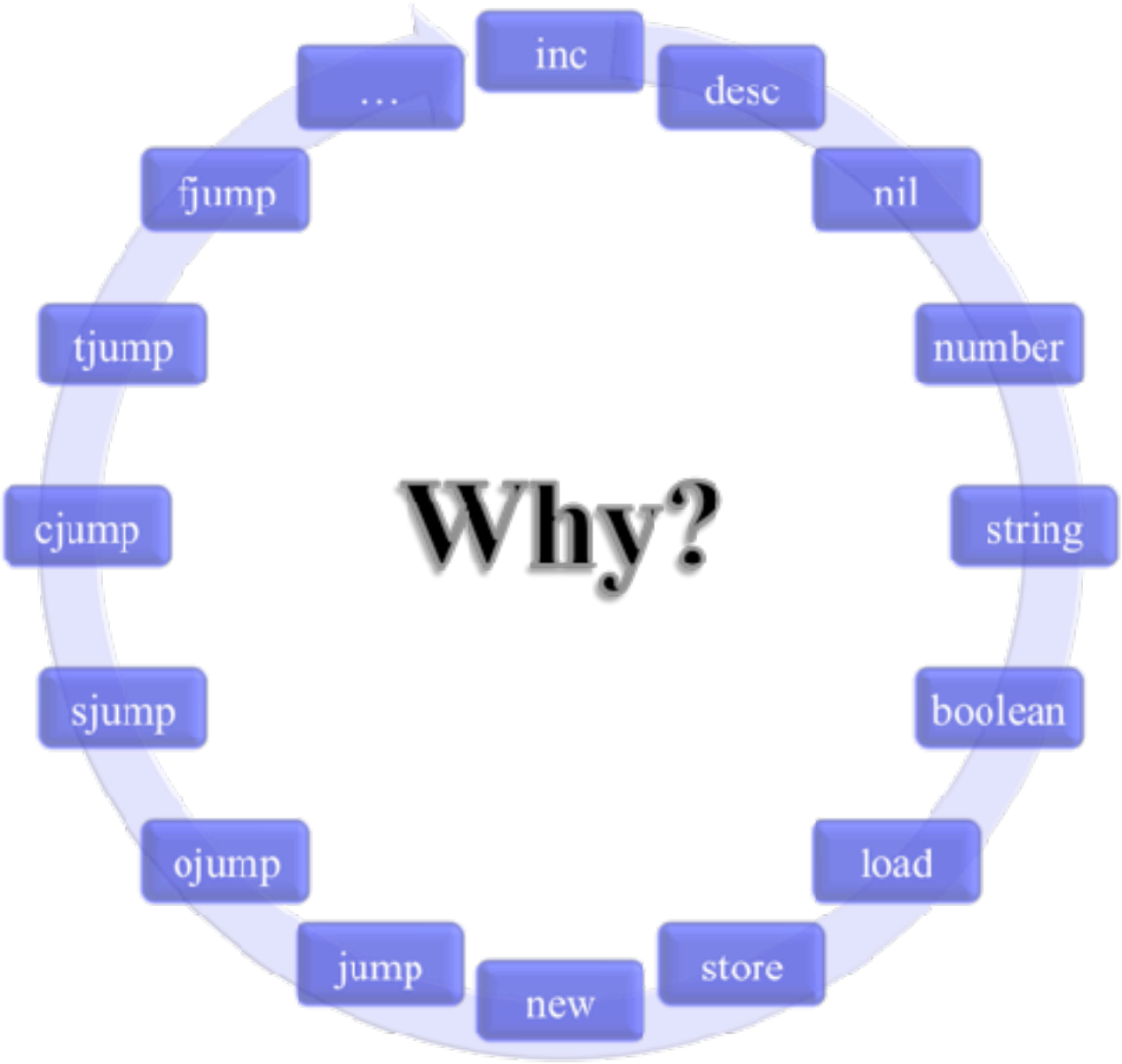
Binary File



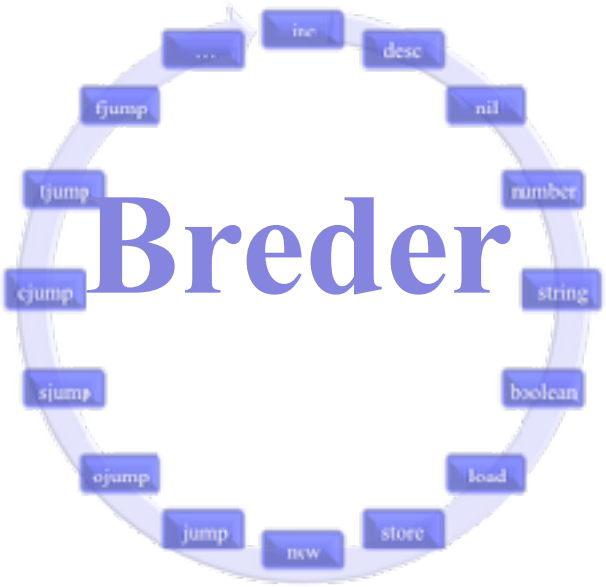
Binary File “binary.b”

Internal File System

Breder Virtual Machine

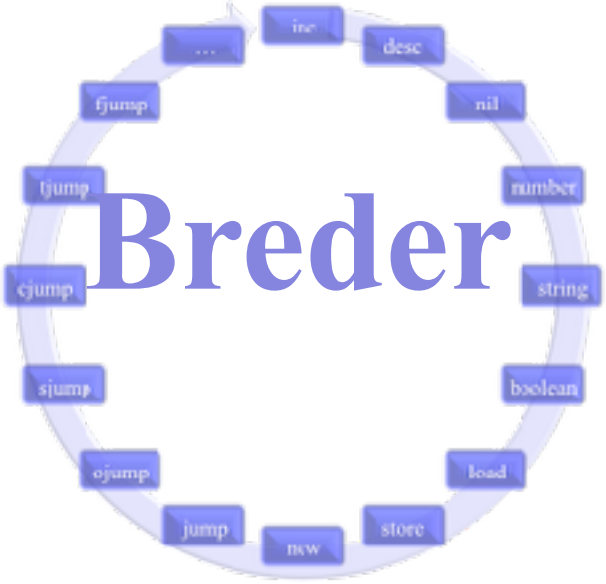


Breder Virtual Machine



Breder

Cuda

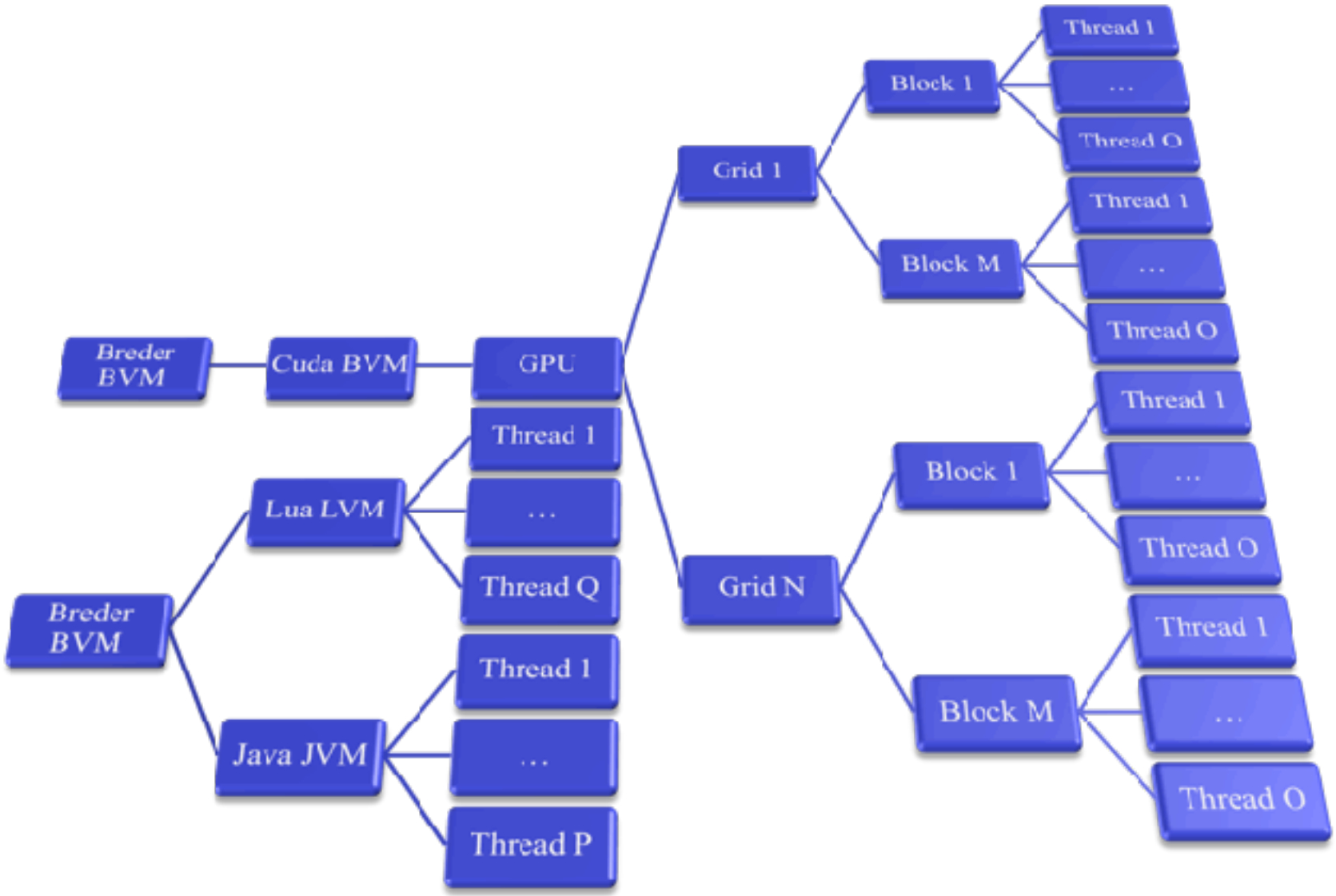


Breder

Java

Lua

Breder Virtual Machine



Garbage Collection

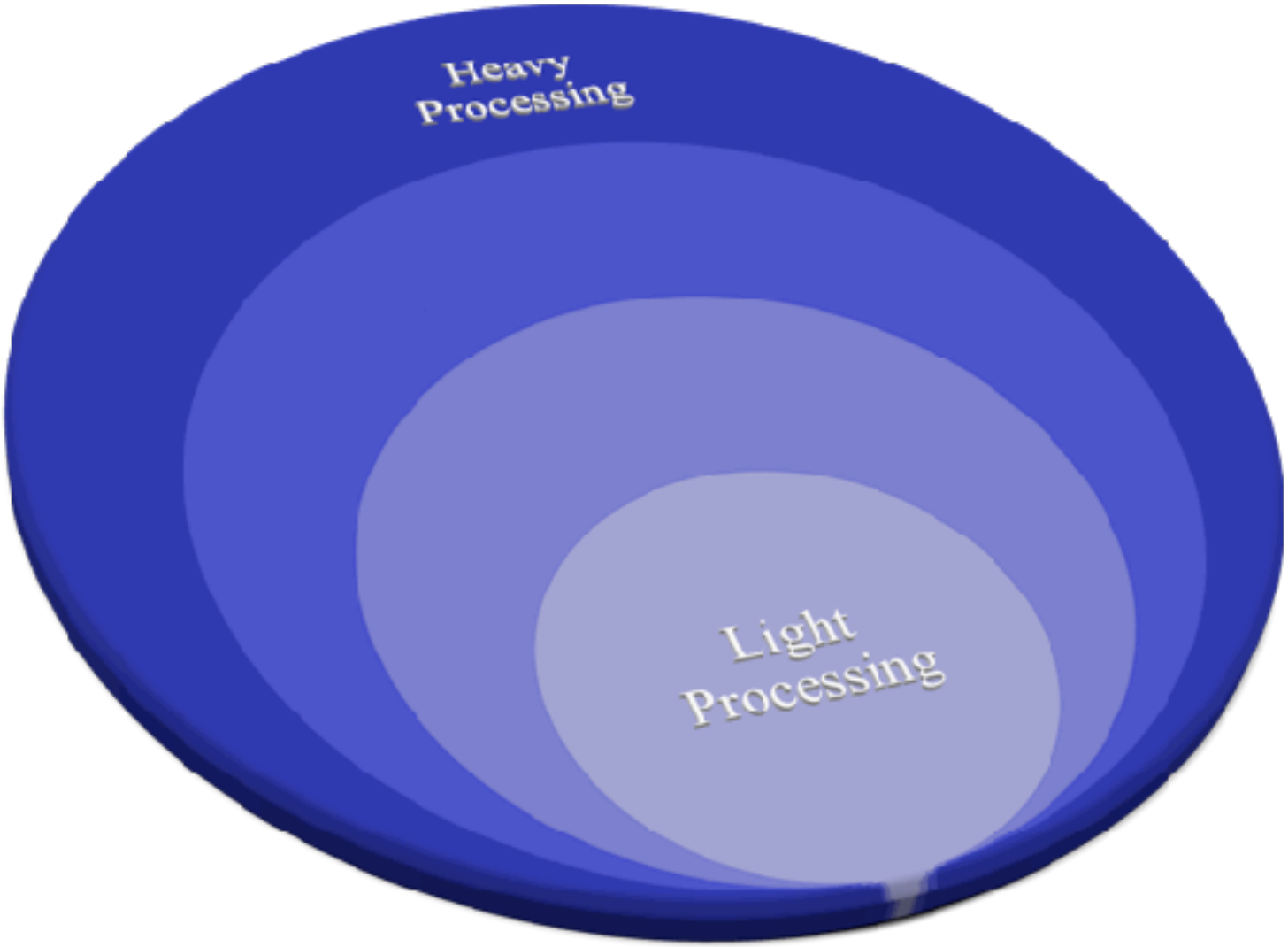
Light Processing

- Coleta a maioria dos Objetos
- Utiliza o Contador de referência

Heavy Processing

- Coleta os Objetos restantes
- Utiliza a Pilha de execução

Garbage Collection

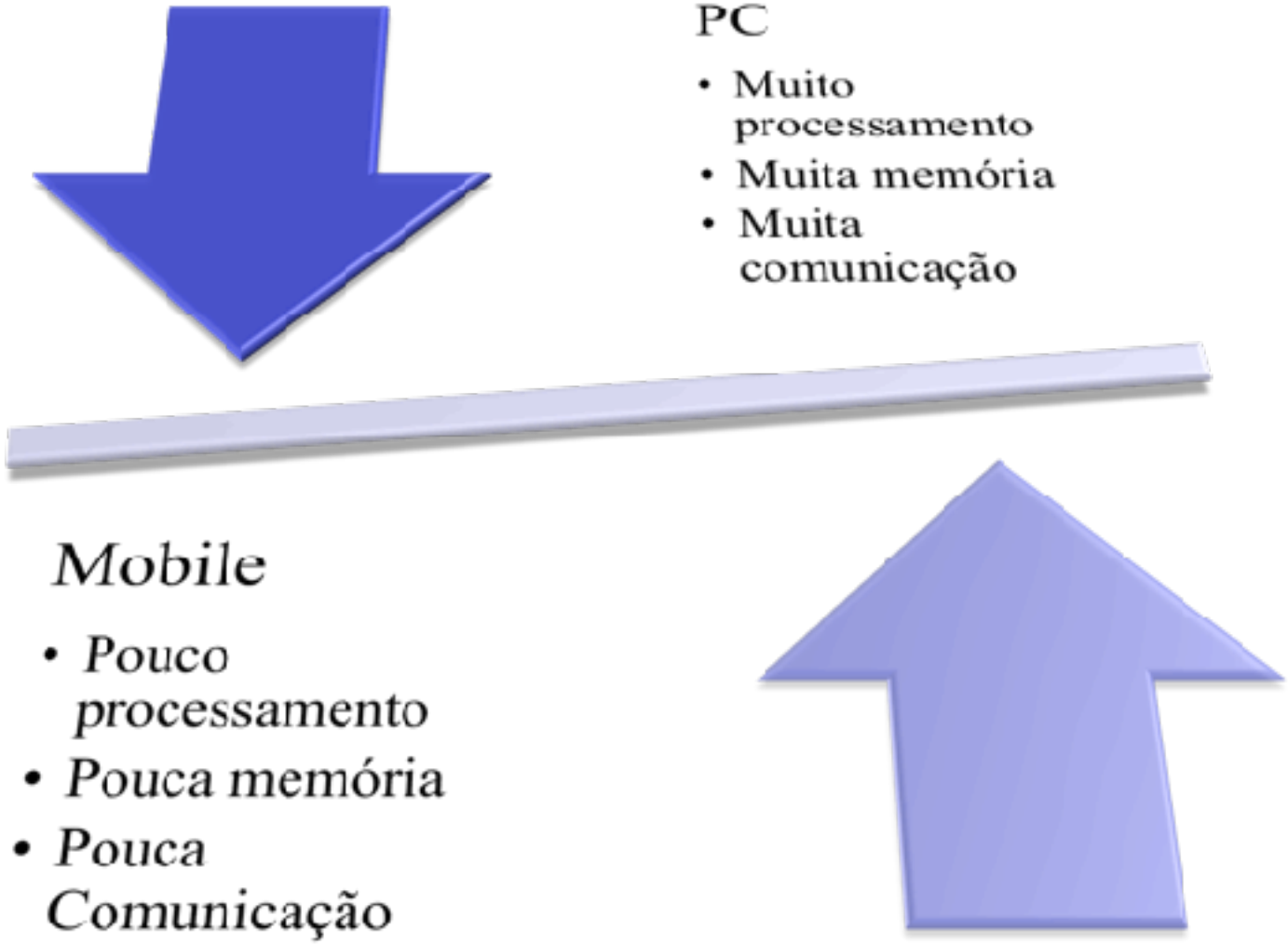


Garbage Collection

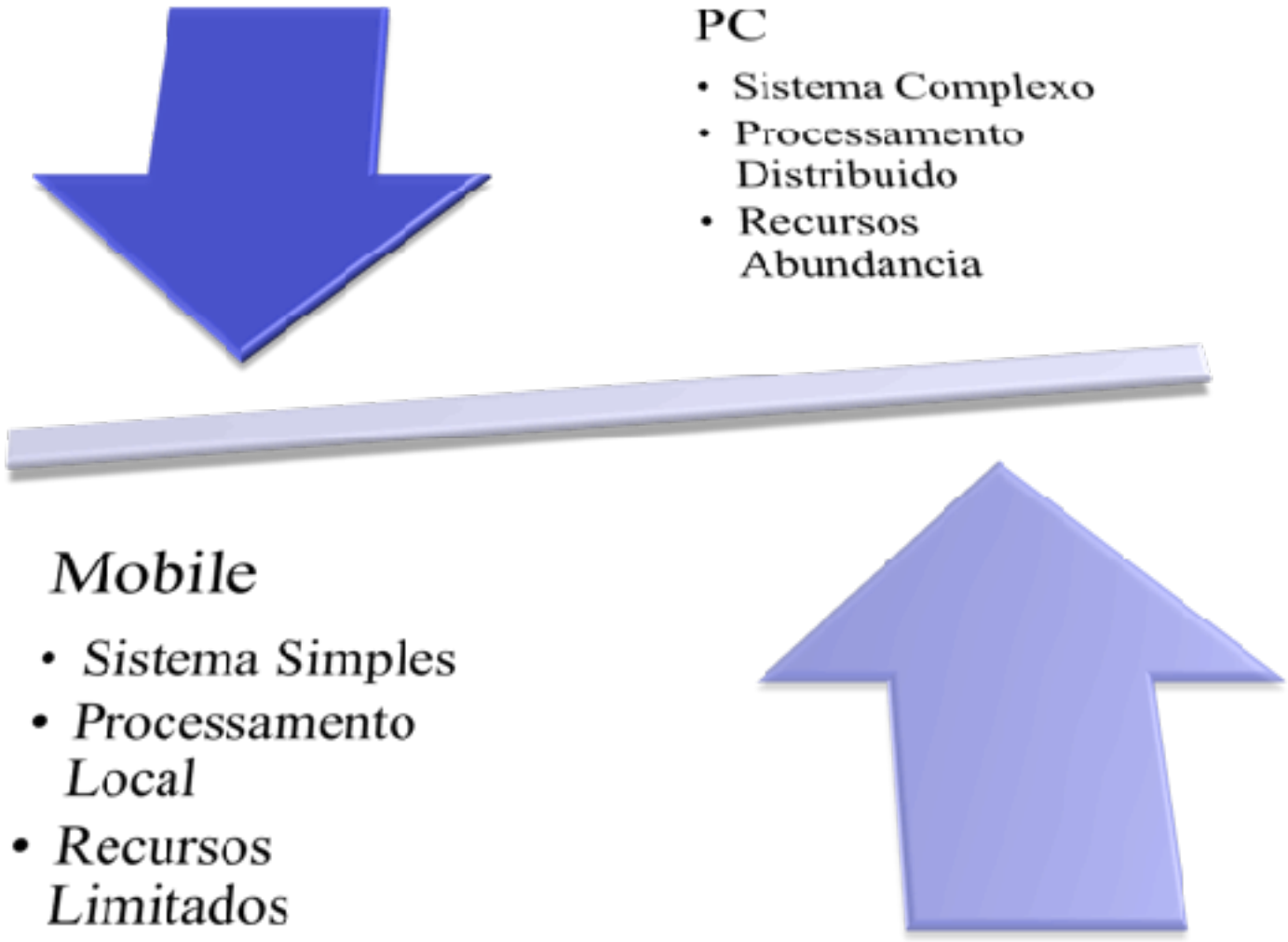
Garbage Collection Light

Garbage Collection Heavy

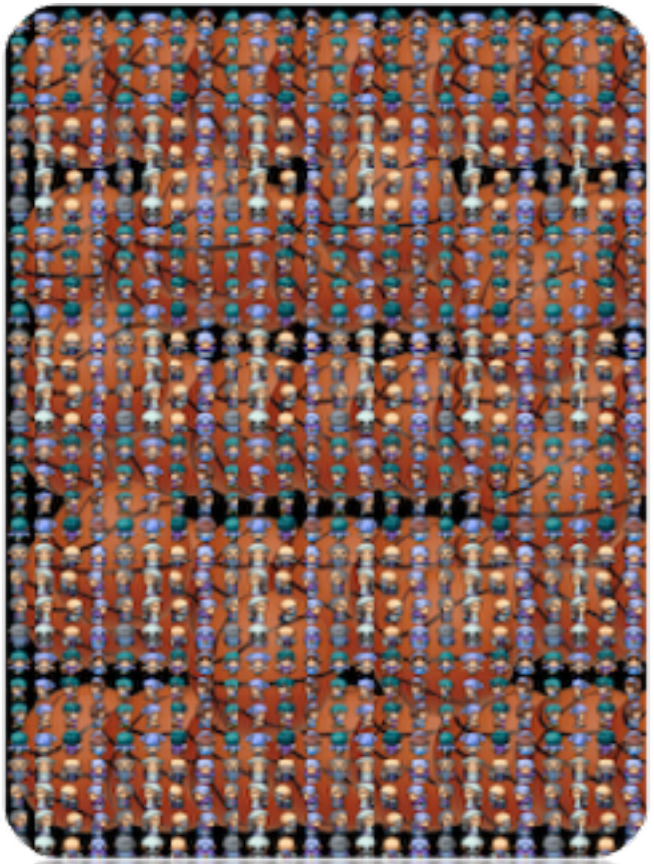
PC x Mobile


PC
• Muito processamento
• Muita memória
• Muita comunicação
Mobile
• Pouco processamento
• Pouca memória
• Pouca Comunicação

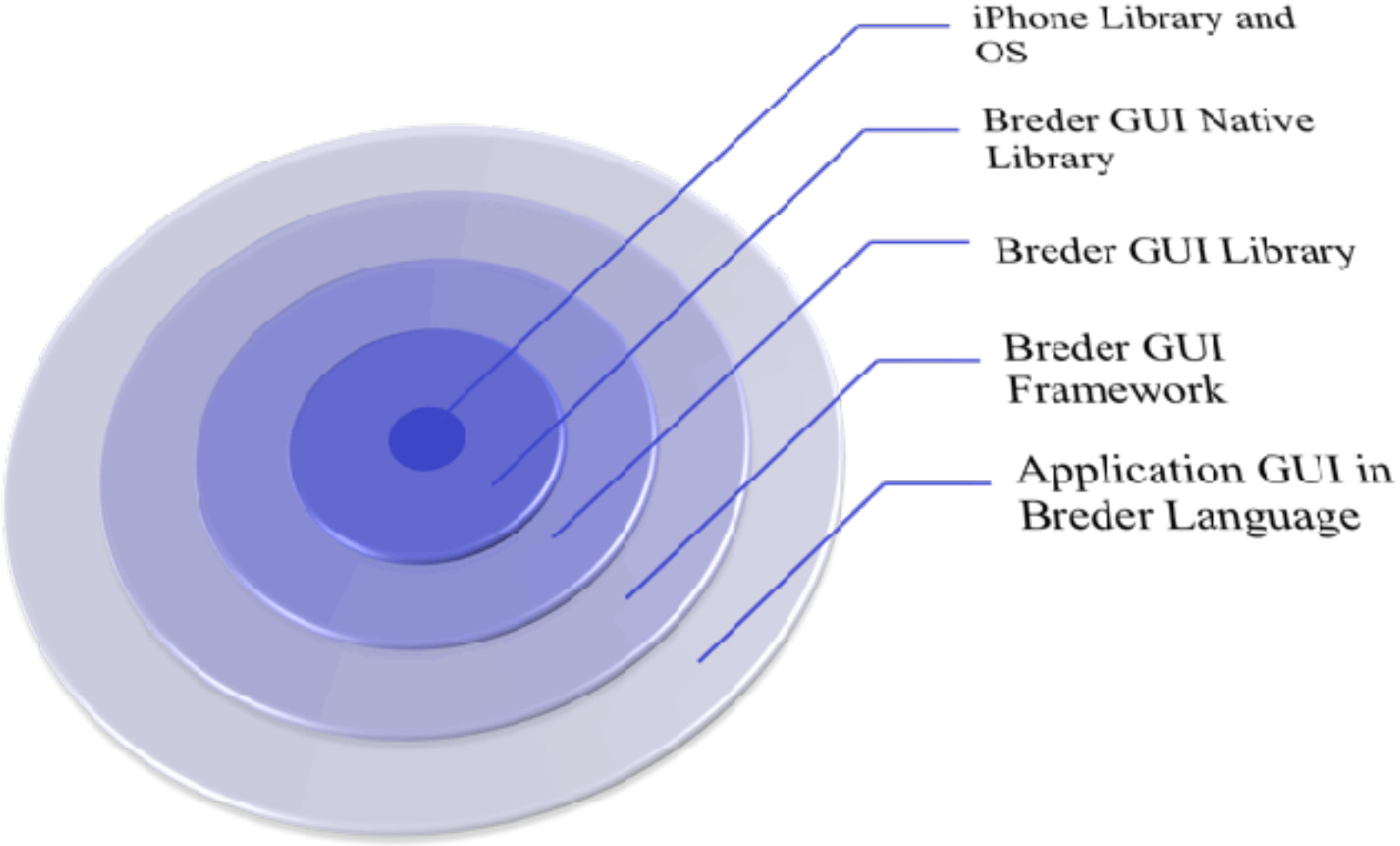
PC x Mobile


PC
• Sistema Complexo
• Processamento Distribuido
• Recursos Abundancia
Mobile
• Sistema Simples
• Processamento Local
• Recursos Limitados

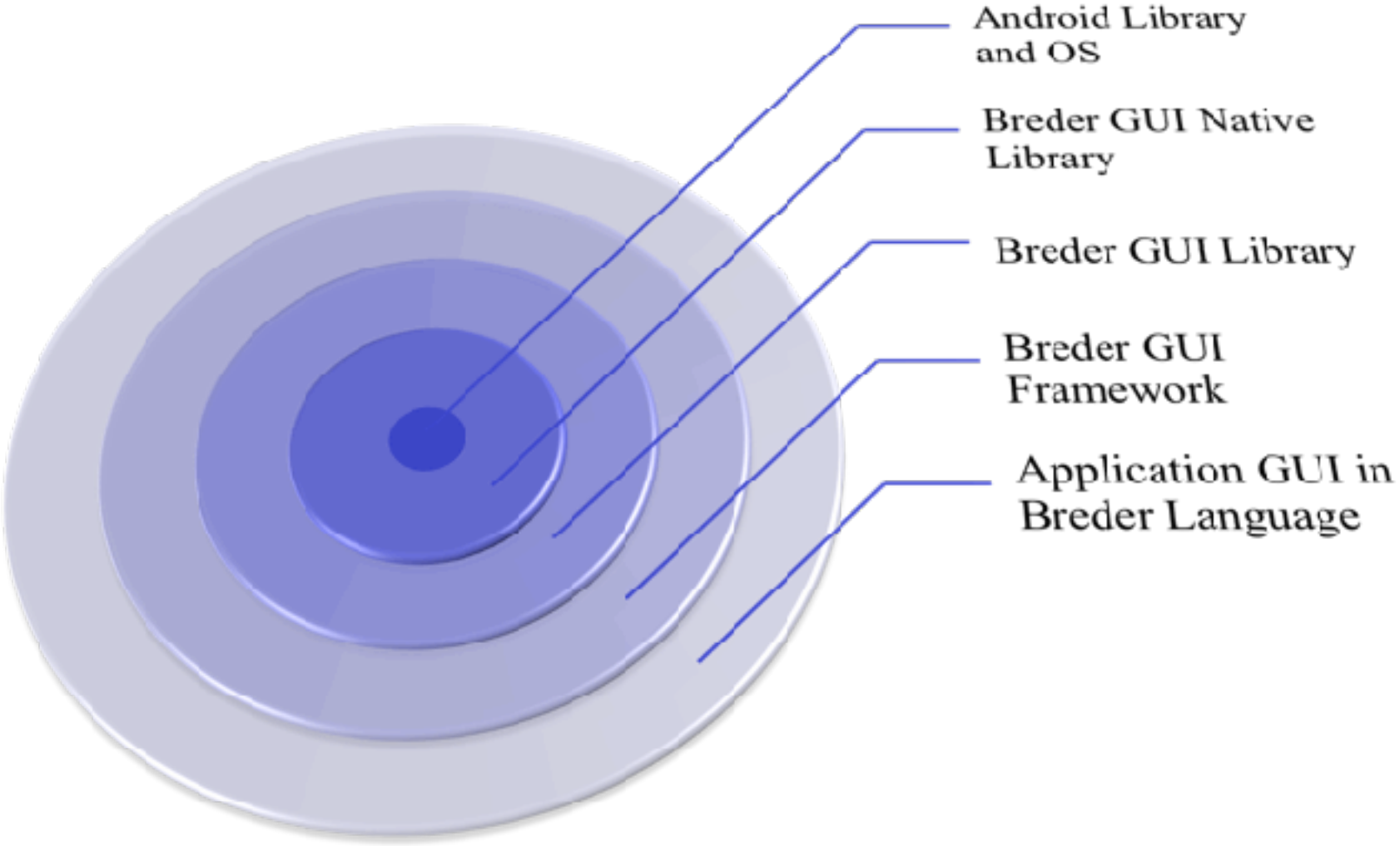
Paper SBGames 2010



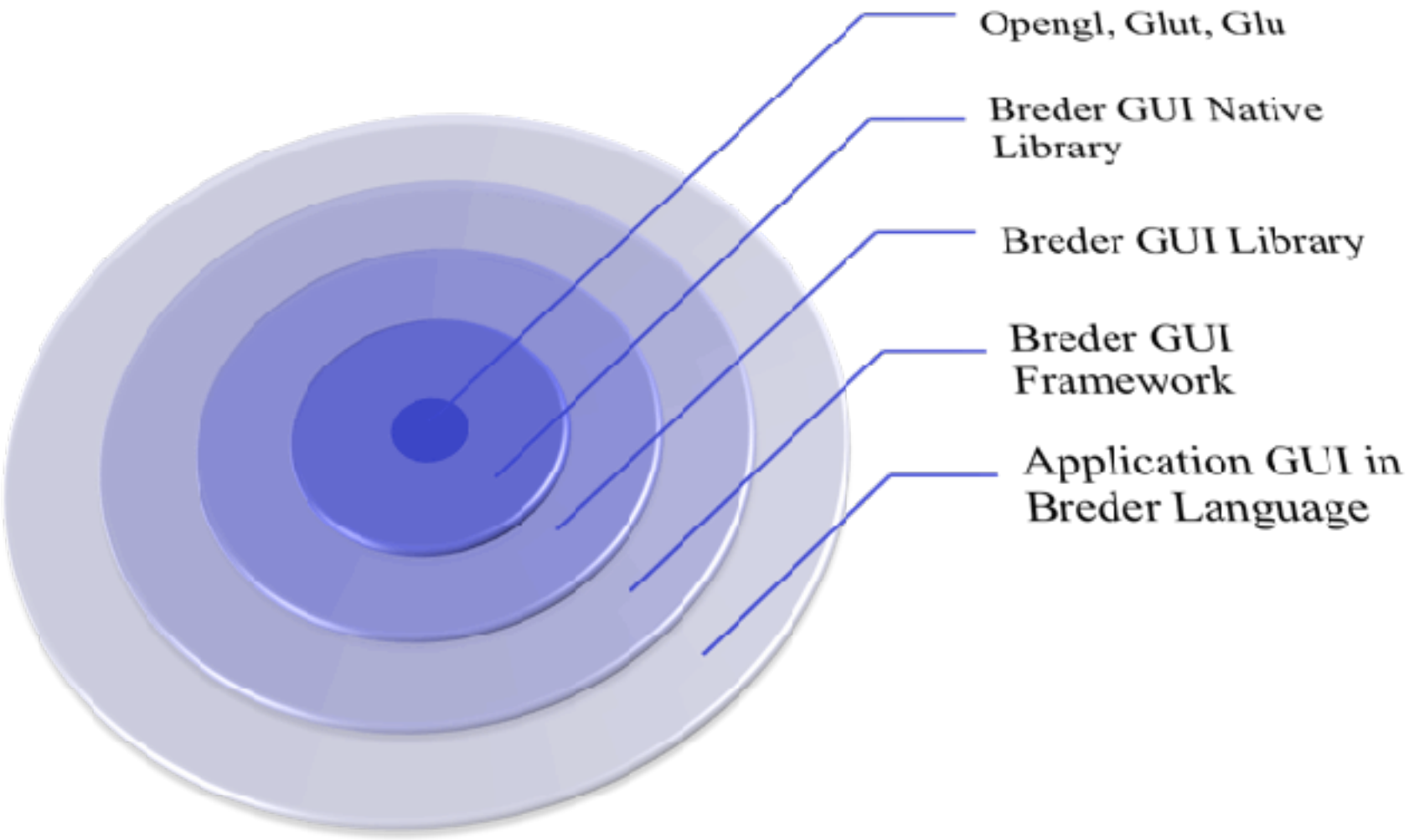
Paper SBGames 2010



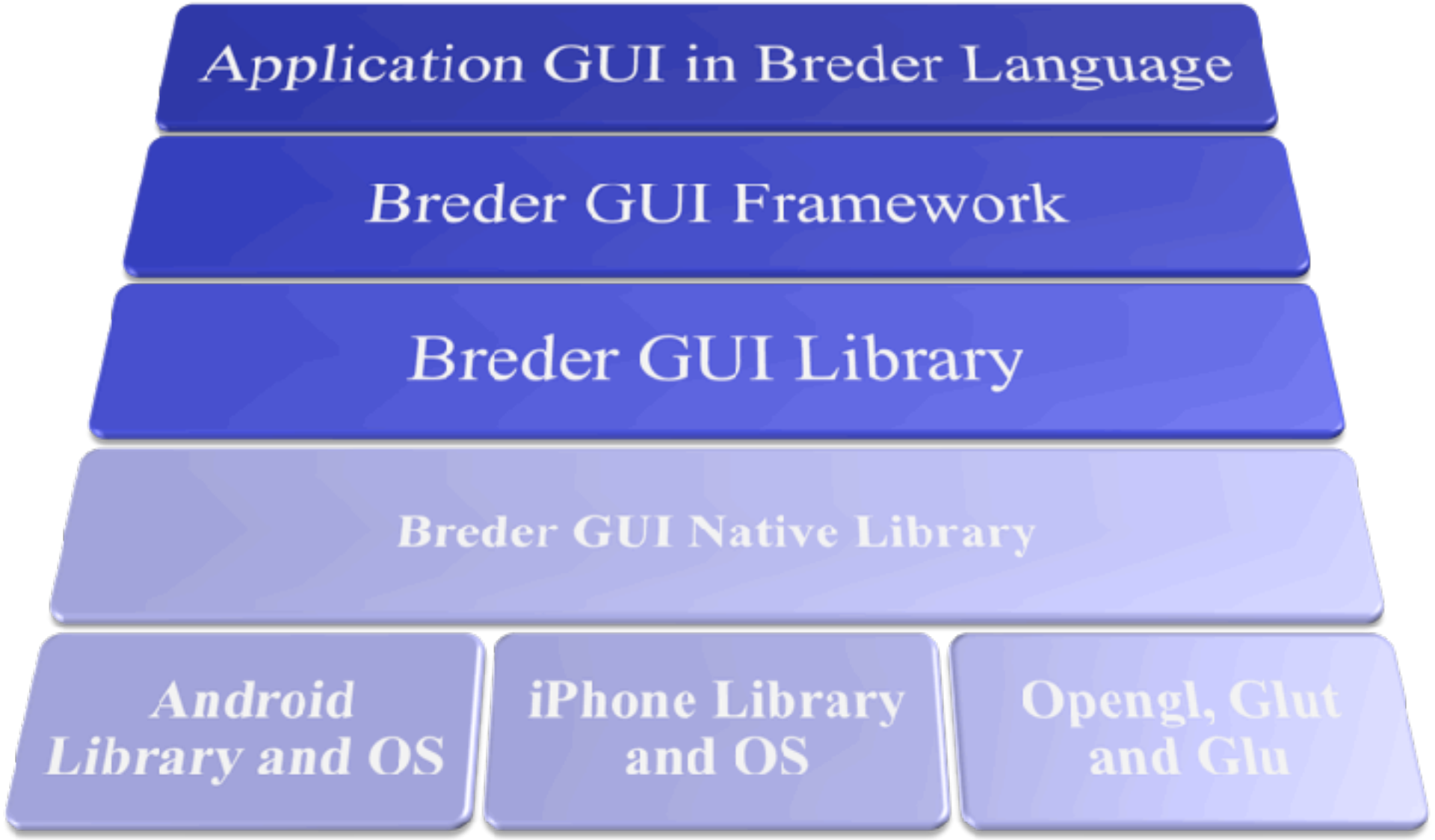
Paper em Andamento



Paper em Andamento



Paper em Andamento



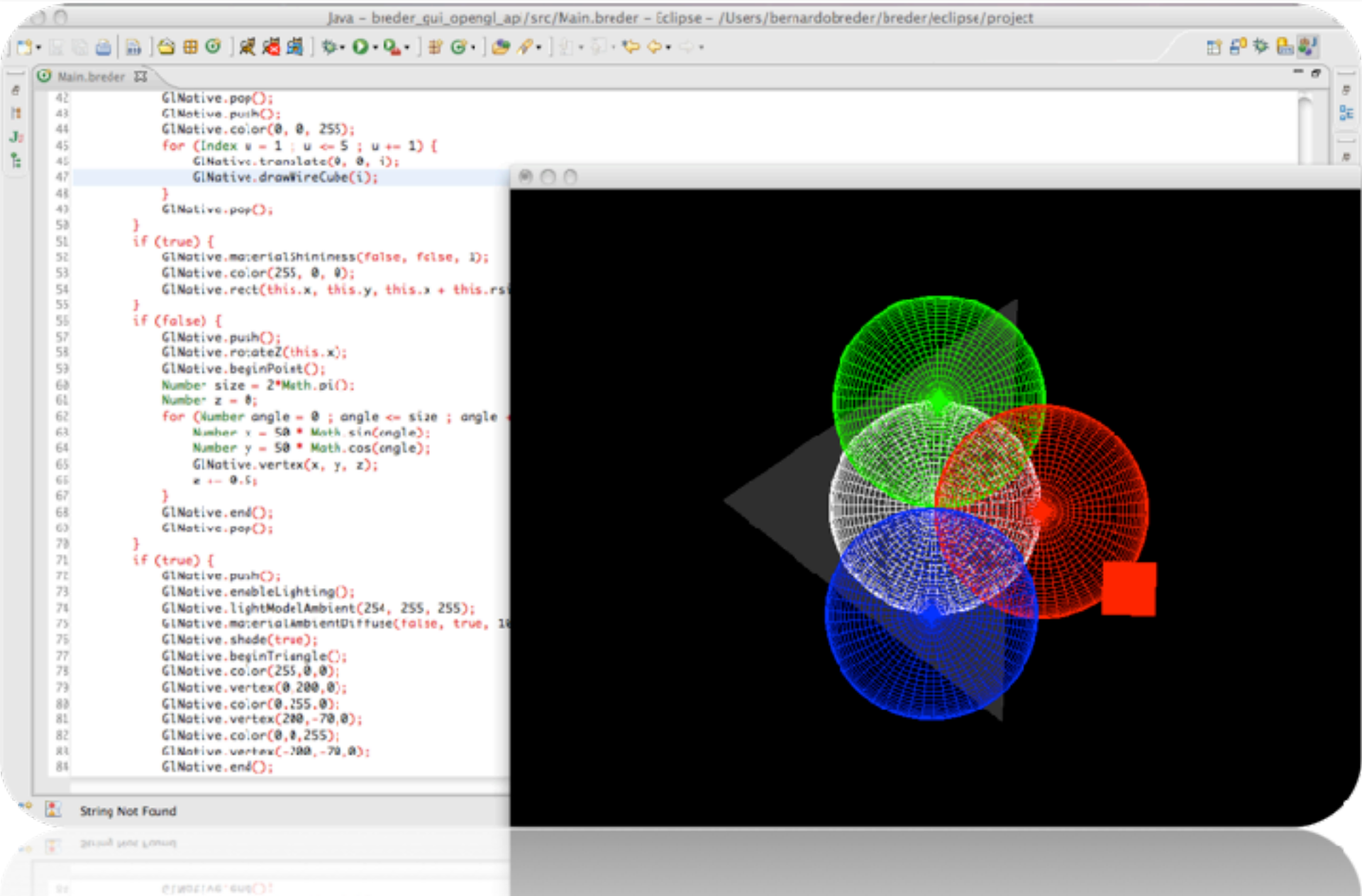
Project


breder compiler
breder vm
breder lang native
breder util native
breder io native
breder gui opengl native
breder sql mysql native
breder sql postgree native
breder lvm
breder jvm
breder cvm
breder processor
breder test
breder profile
breder org lang
breder installer lang

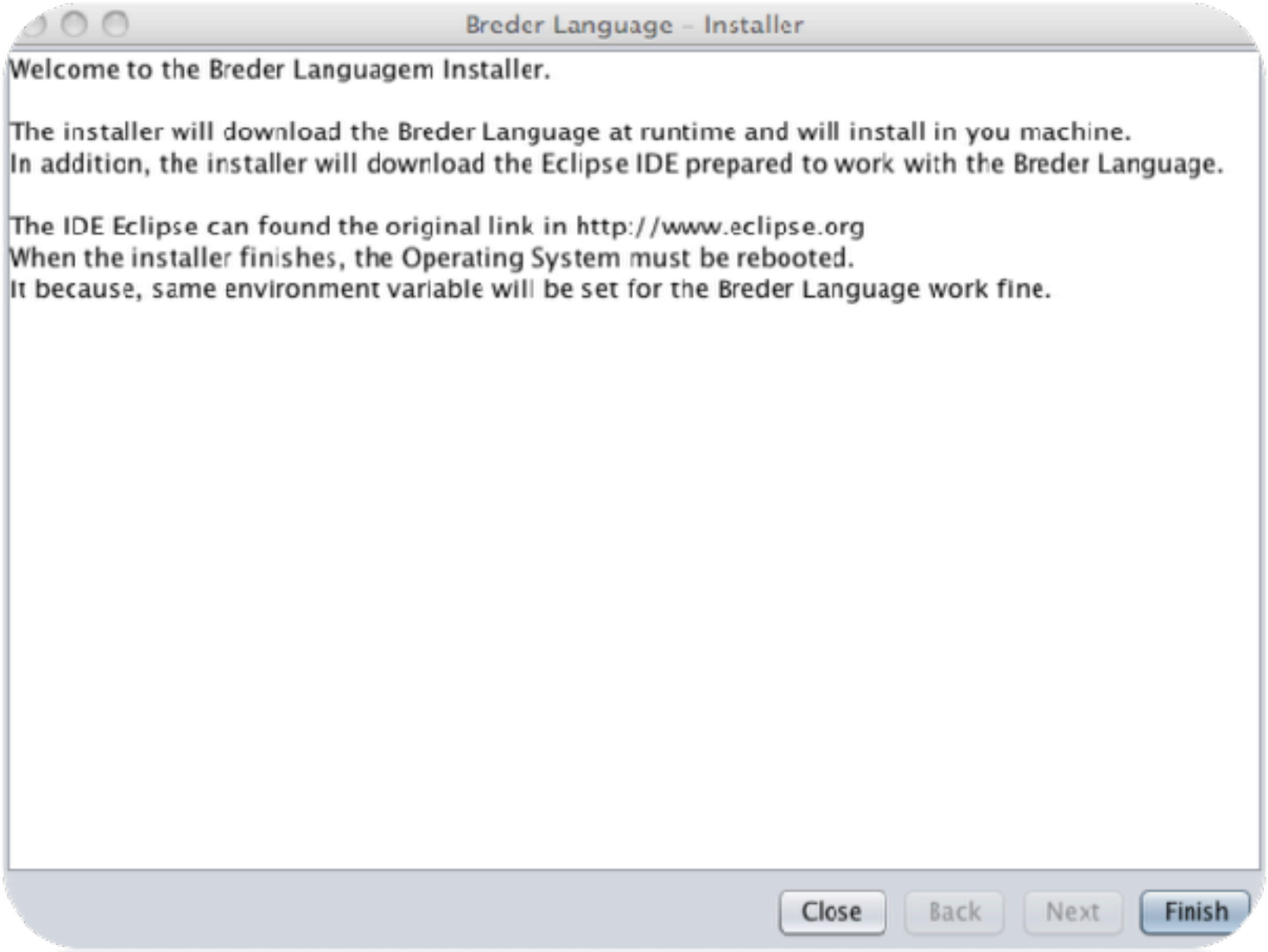
Produtividade

- O Compilador Breder foi desenvolvido em Java, apresentando assim uma grande produtividade no seu crescimento. Assim, toda nova funcionalidade, possui um prazo muito curto para ser desenvolvido.
- Porém, a Máquina Virtual Breder foi implementada na Linguagem C Ansi. Dessa forma, o seu desenvolvimento é bem mais lento, devido a falta de facilidades que a Linguagem possui. Além disso, em certas situações, fica muito difícil deputar a execução.
- A documentação só acompanhou o desenvolvimento, logo após a implementações das estruturas básicas da Linguagem.

Eclipse


Java – breder_gui_opengl_ap/src/Main.breder – Eclipse – /Users/bernardobreder/breder/eclipse/project
Main.breder
GlNative.pop();
GlNative.push();
GlNative.color(0, 0, 255);
for (Index u = 1 ; u <= 5 ; u += 1) {
GlNative.translate(0, 0, i);
GlNative.drawWireCube(i);
}
GlNative.pop();
}
if (true) {
GlNative.materialShininess(false, false, 1);
GlNative.color(255, 0, 0);
}
if (false) {
GlNative.push();
GlNative.rotateZ(this.x);
GlNative.beginPoint();
Number size = 2*Math.pi();
Number z = 0;
for (Number angle = 0 ; angle <= size ; angle
Number x = 50 * Math.sin(angle);
Number y = 50 * Math.cos(angle);
GlNative.vertex(x, y, z);
z += 0.5;
}
GlNative.end();
GlNative.pop();
}
if (true) {
GlNative.push();
GlNative.enableLighting();
GlNative.lightModelAmbient(254, 255, 255);
GlNative.materialAmbientDiffuse(false, true,
GlNative.shade(true);
GlNative.beginTriangle();
GlNative.color(255,0,0);
GlNative.vertex(0,200,0);
GlNative.color(0,255,0);
GlNative.vertex(200,-70,0);
GlNative.color(0,0,255);
GlNative.vertex(-200,-70,0);
GlNative.end();
String Not Found

Installer


Breder Language - Installer
Welcome to the Breder Languagem Installer.
The installer will download the Breder Language at runtime and will install in you machine.
In addition, the installer will download the Eclipse IDE prepared to work with the Breder Language.
The IDE Eclipse can found the original link in http://www.eclipse.org
When the installer finishes, the Operating System must be rebooted.
It because, same environment variable will be set for the Breder Language work fine.
Close
Back
Next
Finish

Hello World

```
[java] : public class HelloWorld {
    public static void main(String[] list) {
      System.out.println("Hello World");
    }
}
[breder] : public class HelloWorld {
    public static void main(IList<String> list) {
      Console.println("Hello World");
    }
}
```

Not Null – Motivação

```
[java] : public class Test {
    public static void main(IList<String> list) {
      name = “test”;
      Integer index = name.indexOf(‘a’);
      name = name.substring(0, index);
    }
}
```

Not Null

```
public class Test {
    public static void main(IList<String> list) {
      Test.test(1);
      Test.test(null);
    }
    public static void test(Number i) {}
}
```

Not Null

```
public class Test {
    public static void main(IList<String> list) {
      Test.test(1);
      Test.test(null); // Fail
    }
    public static void test(notnull Number i) {}
}
```

Not Null

```
public class Test {
    public static void main(IList<String> list) {
      Number i = 1;
      Test.test(i);
      i = null;
      Test.test(i); // Fail
    }
    public static void test(notnull Number i) {}
}
```

Not Null

```
public class Test {
    public static void main(IList<String> list) {
      Number i = 1;
      Test.test1(i);
    }
    public static void test1(notnull Number i) {
      Test.test2(i);
    }
    public static void test2(notnull Number i) {}
}
```

Not Null

```
public class Test {
    public static void main(IList<String> list) {
      Number i = 1;
      Test.test1(i);
    }
    public static void test1(Number i) {
      Test.test2(i); // Fail
    }
    public static void test2(notnull Number i) {}
}
```

Not Null

```
public class Test {
    public static void main(IList<String> list) {
      Number i = test2();
      Test.test1(i);
    }
    public static void test1(notnull Number i) {}
    public static notnull Number test2() { return 1; }
}
```

Not Null

```
public class Test {
    public static void main(IList<String> list) {
      Number i = test2();
      Test.test1(i); // Fail
    }
    public static void test1(notnull Number i) {}
    public static Number test2() { return 1; }
}
```

Not Null

```
public class Test {
    public static void main(IList<String> list) {
      Number i = test2();
      i.toString(); // Fail
    }
    public static Number test2() { return 1; }
}
```

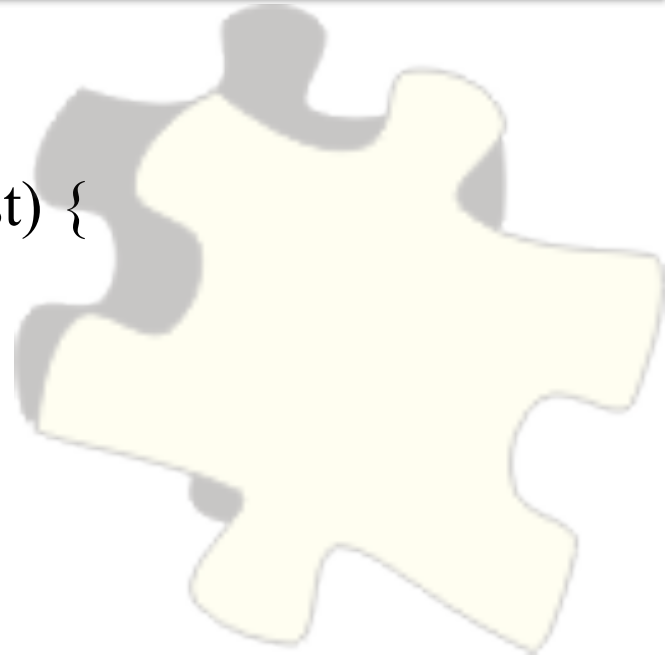
Not Null

```
public class Test {
    public static void main(IList<String> list) {
      Number i = test2();
      if (i != null) { i.toString(); }
    }
    public static Number test2() { return 1; }
}
```

Not Null

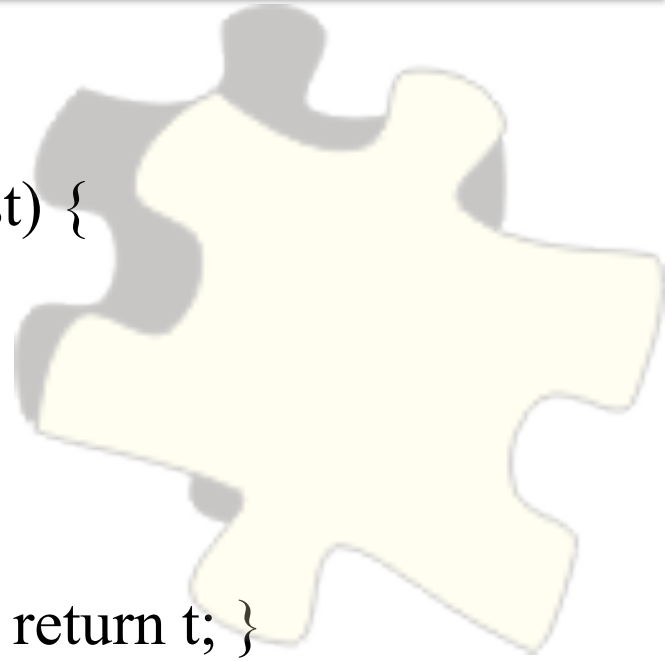
```
public class Test {
    public static void main(IList<String> list) {
      Number i = test2();
      i = (notnull Number) i;
      i.toString();
    }
    public static Number test2() { return 1; }
}
```

Operator



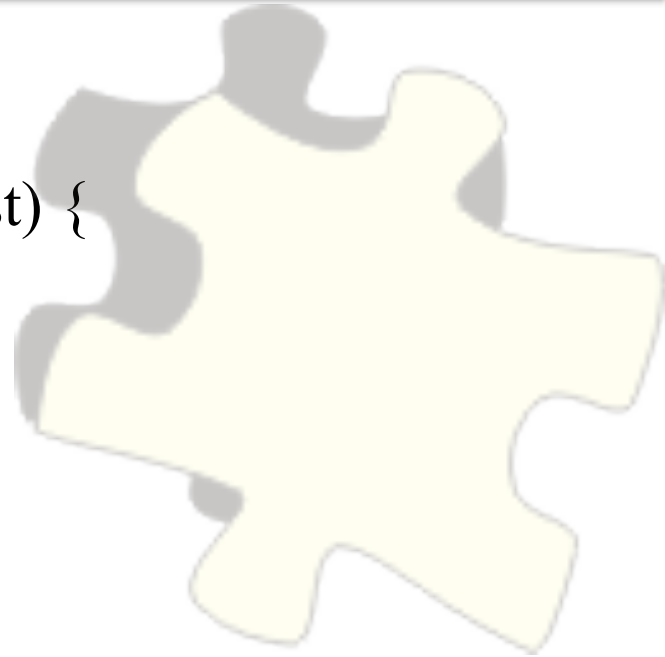
```
public class Test {
    public static void main(IList<String> list) {
        Test t = new Test();
        t = t + t;
    }
    public notnull Test sum(notnull Test t) {
        return t;
    }
}
```

Operator



```
public class Test {
    public static void main(IList<String> list) {
        Test t = new Test();
        t = t + t * t;
    }
    public notnull Test sum(notnull Test t) { return t; }
    public notnull Test mul(notnull Test t) { return t; }
}
```

Operator



```
public class Test {
    public static void main(IList<String> list) {
        Test t = new Test();
        if (t == t) {}
        if (t != t) {}
    }
    public notnull Test equal(notnull Test t) {
        return t;
    }
}
```

Operator

```
public class Test {
    public static void main(IList<String> list) {
        Test t = new Test();
        if (t >= t) {}
    }
    public notnull Boolean highequal(notnull Test t) {
        return true;
    }
}
```

Interval

```
public class Test {
    public static void main(IList<String> list) {
        IInterval i = [1:2];
        IInterval i = [1:2 ; 3];
        IInterval i = [1:2 ; 3 ; 4:5 ] + [6] – [3];
        IInterval i = [1:];
        IInterval i = [1:-1];
        IInterval i = [-2:-1 ; 1];
        IInterval i = [:] – [Math.second(*)];
    }
}
```

Interval

```
public class Test {
    public static void main(IList<String> list) {
        IList<String> l = new ArrayList<String>();
        l.add(5, 10, 15, 20, 25, 30);
        Console.println(l[2:-2]);
        for (Index n : l[[2:-2] + [1] + [-1]]) {}
    }
}
```

Interval

```
public class Test {
    public static void test(Ilist<String> l, IInterval i) {
        for (Index n : l[i]) {}
    }
    public static void main(IList<String> list) {
        Test.test(list, [2:-2] + [1] + [-1]);
        Test.test(list, [:] / [1]);
    }
}
```

Interval

```
public class Test {
    public static void main(IList<String> list) {
        List<Index> a = new ArrayList<Index>().add(5,6,7,8);
        List<Index> b = (a[2,-2] + a[1]) / a[1];
    }
}
```

Dynamic Type

```
public class Test {
    public statitc notnull Index, String test() {
      return 1, null;
    }
    public static void main(IList<String> list) {
      <notnull Index index, String value> i = Test.test();
      Console.println(i.index);
      Console.println(i.value);
    }
}
```

Block Return

```
[java] : List<String> list = (…)
Integer index = null; String value = null;
for (Integer i = 0 ; i < list.size() ; i++) {
    if (list.get(i).equal("test")) {
        index = i;
        value = list.get(i);
        break;
    }
}
```

Block Return

```
IList<String> list = (…)
Index index, String value = {
    for (Index i = 1, list.size()) {
        if (list.get(i) == “test”) {
            breturn i, list.get(i);
        }
    }
}
```

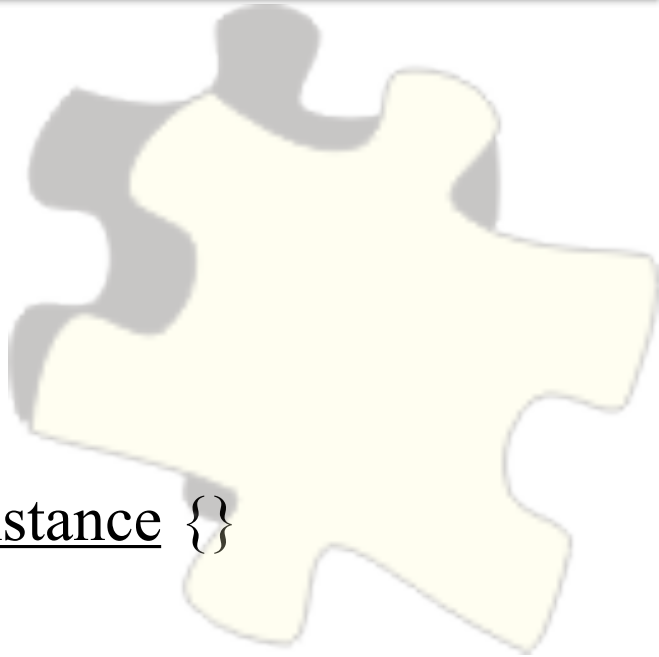
Block Finally

- If {} finally {}
- For (…) {} finally {}
- Repeat (…) {} finally {}
- While (…) {} finally {}
- public void test () {} finally {}
- {} finally {}
- { {} finally {} } finally {}

Singleton

```
public class Test {}
public object Test.instance {
    public void test() {}
}


public static void main(IList<String> list) {
    Test.instance.test();
}
```

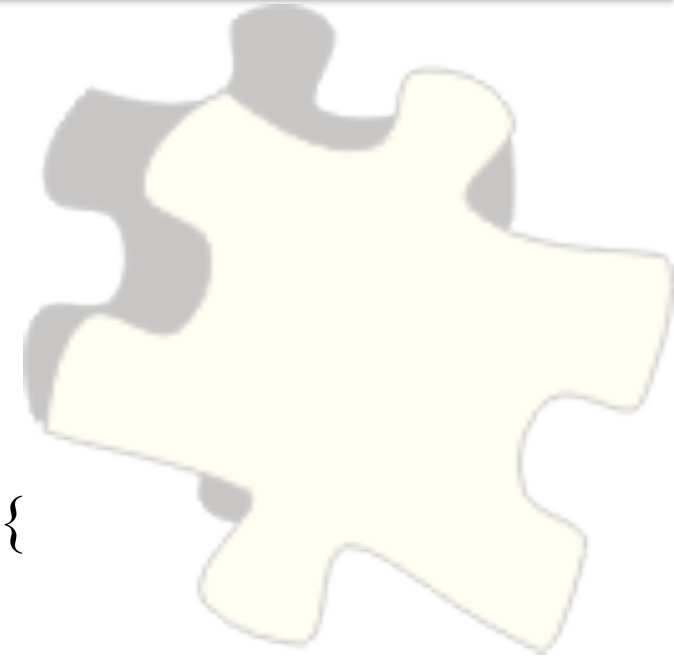
Singleton



```
public class Test {}
public object Test.instance {
    public void test() {}
}
public object Test.instance2 extends Test.instance {}

public static void main(IList<String> list) {
    Test.instance.test();
    Test.instance2.test();
}
```

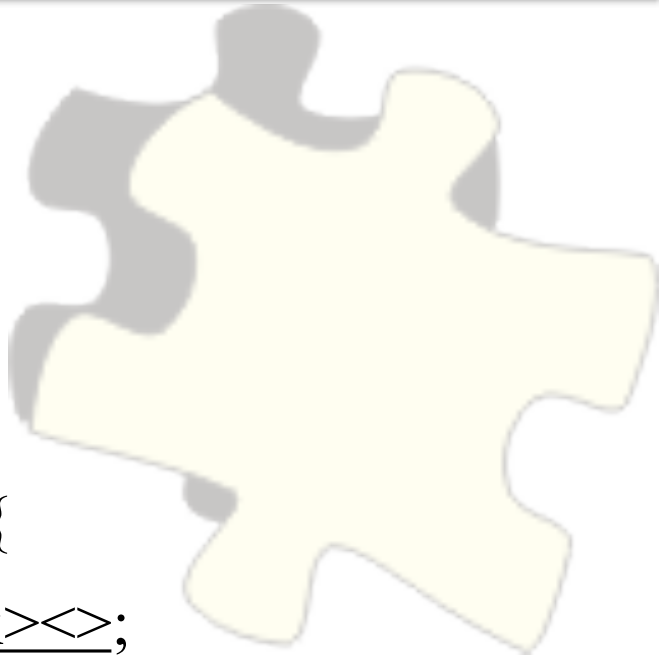
Dynamic Method

```
public class Test {
    public void test() {}
}

public static void main(IList<String> list) {
    Method<><> method = Test.test<><>;
}
```



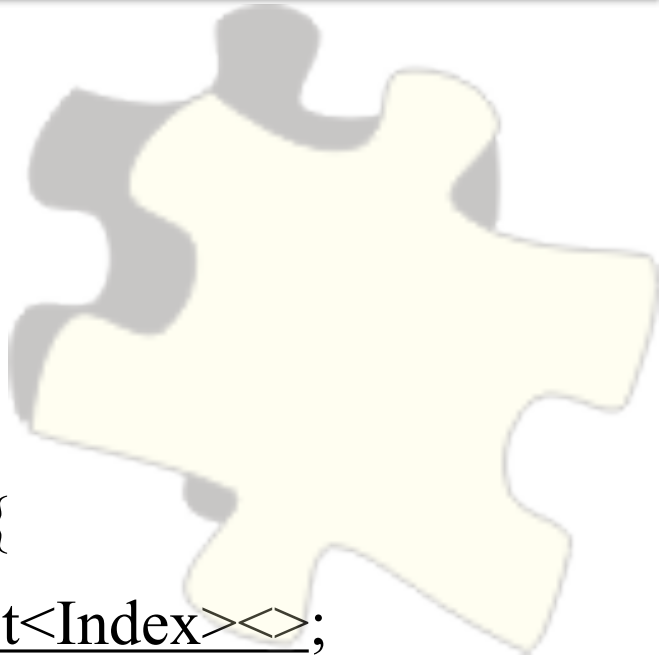
Dynamic Method

```
public class Test {
    public Index test() { return null; }
}

public static void main(IList<String> list) {
    Method<Index><> m = Test.test<Index><>;
}
```



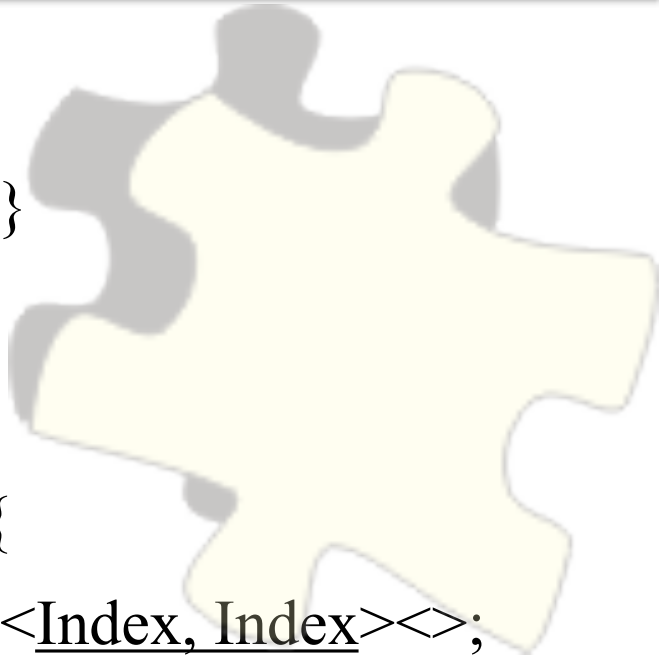
Dynamic Method

```
public class Test {
    public notnull Index test() { return 1; }
}

public static void main(IList<String> list) {
    Method<notnull Index><> m = Test.test<Index><>;
}
```



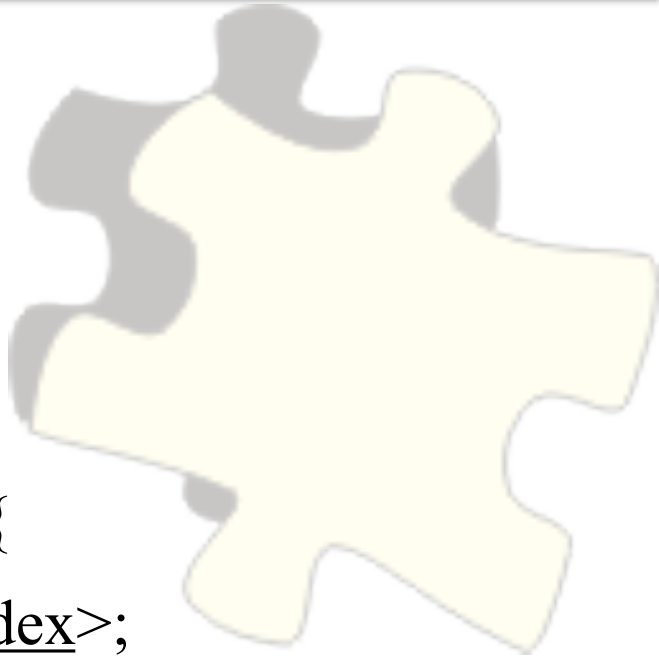
Dynamic Method

```
public class Test {
    public Index, Index test() { return 1, 2; }
}

public static void main(IList<String> list) {
    Method<Index, Index><> m = Test.test<Index, Index><>;
}
```



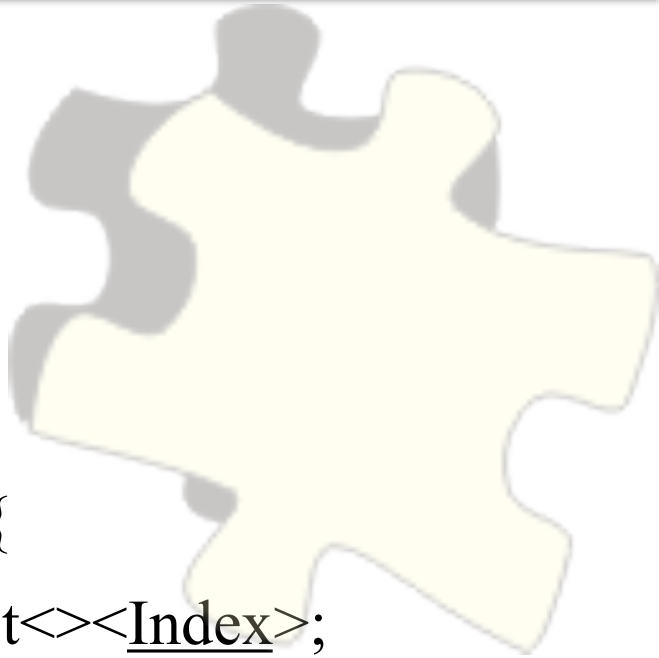
Dynamic Method

```
public class Test {
    public void test(Index i) {}
}

public static void main(IList<String> list) {
    Method<><Index> m = Test.test<><Index>;
}
```



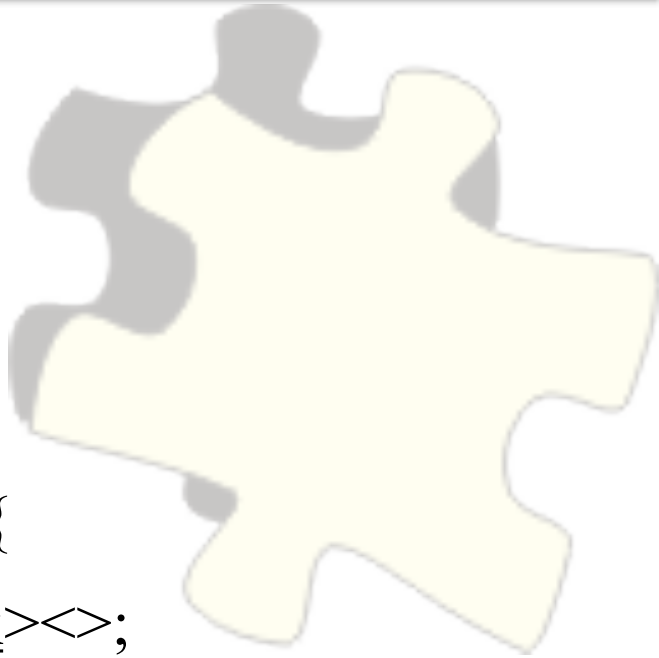
Dynamic Method

```
public class Test {
    public void test(notnull Index i) {}
}

public static void main(IList<String> list) {
    Method<><notnull Index> m = Test.test<><Index>;
}
```



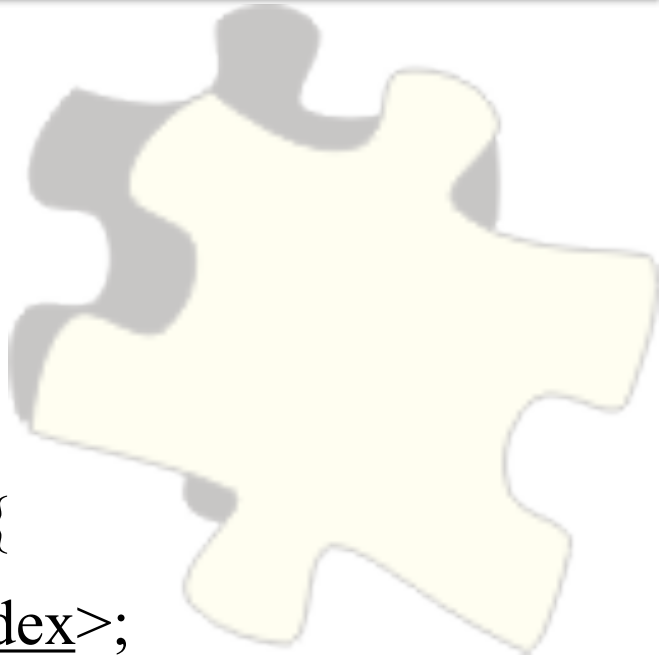
Dynamic Method

```
public class Test {
    public Index test() { return null; }
}

public static void main(IList<String> list) {
    Method<Index><> m = Test.test<Index><>;
    Index i = m();
}
```



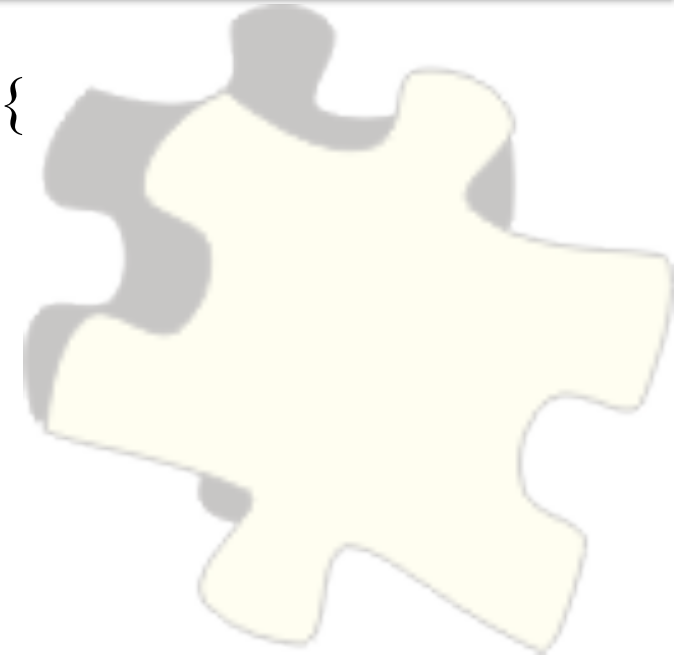
Dynamic Method

```
public class Test {
    public void test(Index i) {}
}

public static void main(IList<String> list) {
    Method<><Index> m = Test.test<><Index>;
    m(1);
}
```



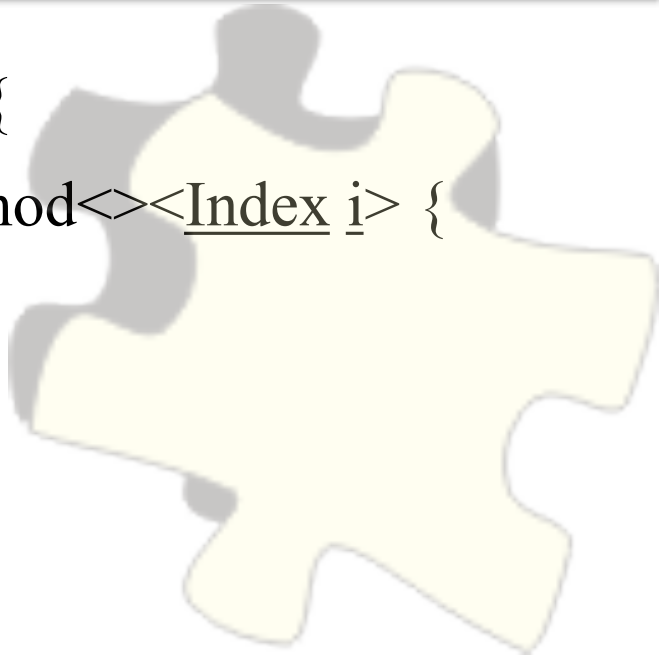
Dynamic Method

```
public static void main(IList<String> list) {
    new Method<notnull Index><> {
        return 1;
    }();
}
```



Dynamic Method

```
public static void main(IList<String> list) {
    Method<><Index> method = new Method<><Index i> {
        Console.println(i);
    }
    method(5);
}
```



Dynamic Method

```
public class Test {
    public void test (Index i) { Console.println(“old”); }
}
public static void main(IList<String> list) {
    Test t = new Test();
    t.test<><> = new Method<><Index i> {
            Console.println(“new”);
    }
    t.test();
}
```

Injection of Method

```
public plugin String {
    public String[] myDiv() {
        return null;
    }
}

public static void main(IList<String> list) {
    String[] str = “test”.myDiv();
}
```

Injection of Method

```
public plugin String {
    public String toString() { return null; } // Fail
}
```

Injection of Method

[My.bar] :

```
public plugin String {
    public String[] myDiv() { return null; }
}
```

[Other.bar] :

```
public static void main(IList<String> list) {
    String[] str = “test”.myDiv(); // Fail
}
```

Field of Method

```
public notnull Index test () {
    private Index i;
    if (i == null) { i = 1; }
    return this.i;
}

public notnull Index test () {
    private notnull Index i = 1;
    return this.i;
}
```

C Together

[c] : int xyz_Test$print(b_vm_t*) {

printf(“test”);

return B_BNI_SUCESS;

}

[breder] : package xyz;

public class Test {

public native void print();

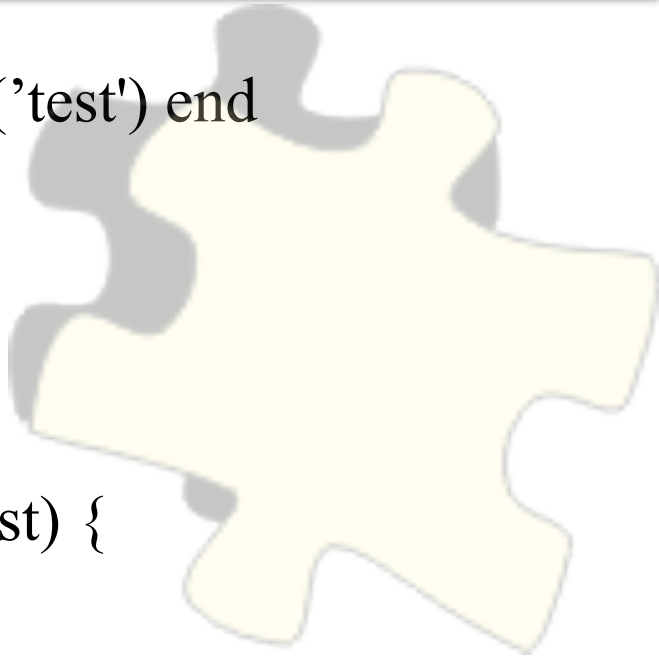
public static void main(IList<String> list) {

new Test().println();

}

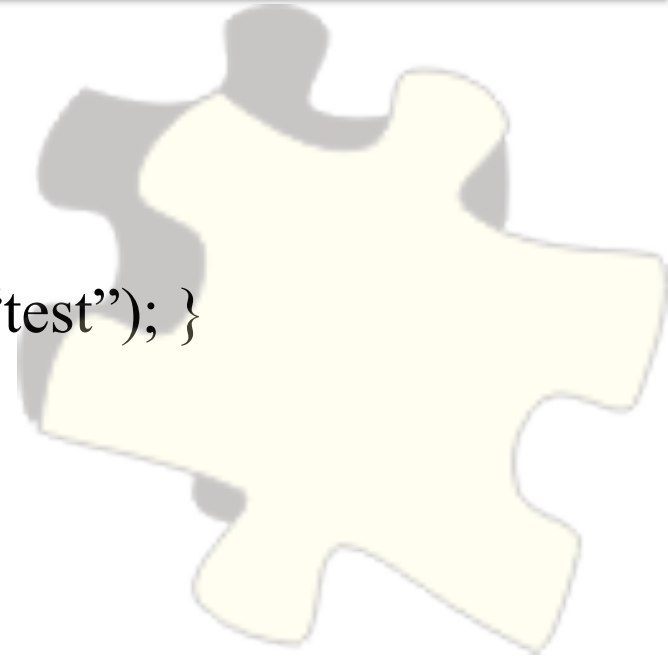
}

Lua Together



```
[lua]  : function xyz_Test$print(bvm) print('test') end
[breder] : package xyz;
public class Test {
    public lua void print();
    public static void main(IList<String> list) {
        new Test().println();
    }
}
```

Java Together



```
[java]  : package xyz;
public class Test {
    public void print() { System.out.print("test"); }
}
[breder] : package xyz;
public class Test {
    public java void print();
    public static void main(IList<String> list) {
        new Test().println();
    }
}
```

Cuda Together

```
[breder] : public class Test {
    public cuda void print(notnull Natural grid, notnull Natural
  block, notnull Natural thread, notnull Natural size) {
            Natural i = block * size + thread;
            if (thread > 1) this.print(grid, block, thread-1);
    }
    public static void main(IList<String> list) {
            new Test().print(2, 3, 4, 10);
    }
}
```

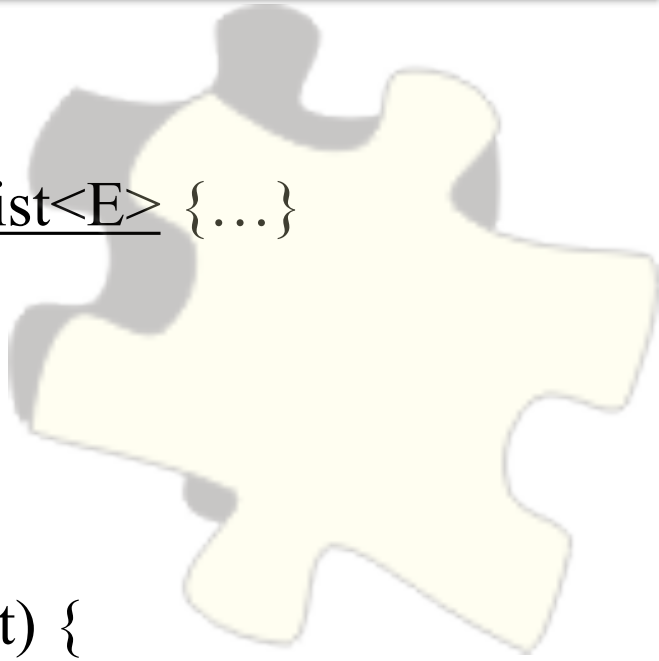
Update Object

```
public class TestA {}
public class TestB extends TestA {}
public class TestC extends TestB {}
public static void main(IList<String> list) {
    TestA ta = new TestA();
    TestB tb = update TestB(ta)();
    TestC tc = update TestC(ta)();
    TestC tcc = update TestC(tb)();
}
```

Hierarchy Package

```
package breder.util;
public class ArrayList<E> implements IList<E> {…}


package xyz;
Public class XYZ {
    public static void main(IList<String> list) {
     ArrayList<Number> list = new ArrayList<Number>();
    }
}
```

Hierarchy Package



```
package breder.util;
public class ArrayList<E> implements *IList<E> {…}


package xyz;
Public class XYZ {
    public static void main(IList<String> list) {
     IList<Number> list = new ArrayList<Number>();
    }
}
```

Hierarchy Package

```
package breder.util;
public class ArrayList<E> implements *IList<E>, IMap<E,E>
    {…}


package xyz;
Public class XYZ {
    public static void main(IList<String> list) {
     IList<Number> list = new ArrayList<Number>();
    }
}
```

Hierarchy Package

```
package breder.util;
public class ArrayList<E> implements *IList<E>, IMap<E,E>
    {…}

package breder.util;
Public class XYZ {
    public static void main(IList<String> list) {
     IMap<Number,Number> list = new ArrayList<Number>();
    }
}
```

Demo

- Ambiente de Teste
- Ambiente de Depuração
- Framework Gráfico
- Installer
- Hello World

Obrigado