

Construção de um compilador de Python para Dalvik usando Objective Caml

Miguel Henrique de Brito Pereira

miguelhbrito@gmail.com

Faculdade de Computação
Universidade Federal de Uberlândia

19 de março de 2019

Lista de Figuras

2.1	ADV Manager	10
2.2	Criar um novo device	11
2.3	Modelo Device	11
2.4	Manager Devices	12

Lista de Tabelas

Lista de Listagens

3.1	Nano01.py	13
3.2	Nano01.java	13
3.3	Nano01.smali	13
3.4	Nano02.py	14
3.5	Nano02.java	14
3.6	Nano02.smali	14
3.7	Nano03.py	14
3.8	Nano03.java	15
3.9	Nano03.smali	15
3.10	Nano04.py	15
3.11	Nano04.java	15
3.12	Nano04.smali	16
3.13	Nano05.py	16
3.14	Nano05.java	16
3.15	Nano05.smali	16
3.16	Nano06.py	17
3.17	Nano06.java	17
3.18	Nano06.smali	17
3.19	Nano07.py	18
3.20	Nano07.java	18
3.21	Nano07.smali	18
3.22	Nano08.py	19
3.23	Nano08.java	19
3.24	Nano08.smali	19
3.25	Nano09.py	20
3.26	Nano09.java	20
3.27	Nano09.smali	21
3.28	Nano10.py	21
3.29	Nano10.java	22
3.30	Nano10.smali	22
3.31	Nano11.py	22
3.32	Nano11.java	23
3.33	Nano11.smali	23
3.34	Nano12.py	24
3.35	Nano12.java	24
3.36	Nano12.smali	24

Sumário

Lista de Figuras	2
Lista de Tabelas	3
1 Introdução	6
1.1 Sistema Operacional	6
1.2 Python	6
1.3 Dalvik	6
1.4 Smali/Baksmali	7
1.5 OCaml	7
2 Instalações	8
2.1 Python	8
2.2 Java	8
2.3 Dalvik	8
2.4 OCaml	9
2.5 Compilação	9
2.5.1 Compilando Java em .dex	10
2.5.2 Complilando .dex em .smali	10
2.5.3 Compilando .smali code em .dex	10
2.6 Executando o arquivo .dex no Android	10
2.6.1 Utilizando Emulador AVD	10
3 Nano Programas	13
3.1 Nano01	13
3.2 Nano02	14
3.3 Nano03	14
3.4 Nano04	15
3.5 Nano05	16
3.6 Nano06	17
3.7 Nano07	18
3.8 Nano08	19
3.9 Nano09	20
3.10 Nano10	21
3.11 Nano11	22
3.12 Nano12	24
4 Referências	26

Capítulo 1

Introdução

Este documento foi escrito para auxiliar na confecção do relatório da disciplina de Construção de Compiladores com a finalidade de detalhar todo o trabalho desenvolvido e os processos envolvidos da Construção de um Compilador, mais especificamente, um Compilador de Python para Dalvik, utilizando a linguagem OCaml para a construção do mesmo.

1.1 Sistema Operacional

Para esse trabalho foi utilizado o sistema operacional *Fedora 28*, sua instalação é fácil e rápida, basta acessar o [site](#) e seguir os passos descritos pela desenvolvedora do sistema.

1.2 Python

Python é uma ótima linguagem de programação orientada a objetos, interpretada e interativa é uma linguagem de programação orientada a objetos, interpretada, de script, interativa, funcional e de tipagem dinâmica. Criada por Guido van Rossum em 1991, hoje segue o modelo de desenvolvimento comunitário, aberto e gerenciado pela organização sem fins lucrativos *Python Software Foundation*.

1.3 Dalvik

Desenvolvida por Dan Bornstein e com contribuições de outros engenheiros do Google, é uma máquina virtual baseada em registradores e foi projetada para ser utilizada no sistema operacional Android. É muito conhecida pelo seu bom desempenho, pelo baixo consumo de memória e foi projetada para permitir que múltiplas instâncias da máquina virtual rodem ao mesmo tempo. A *Dalvik* é frequentemente confundida com uma Java Virtual Machine, porém, o bytecode que ela opera é bastante diferente do bytecode da JVM. A VM do Dalvik, executa um bytecode no formato `.dex` (Dalvik Executable), codigos em `.dex`

são ilegíveis aos humanos, portanto, neste trabalho usaremos Smali Code para apresentar os códigos.

1.4 Smali/Baksmali

O *smali/baksmali* é um *assembler/disassembler* para o formato *.dex* usado pela *Dalvik*, que gera um arquivo *SmaliCode*. A sintaxe é vagamente baseada na sintaxe do *Jasmin*, e suporta a funcionalidades do formato *.dex* (anotações, informações de depuração, informações de linha, etc.)

1.5 OCaml

Objective Caml, ou somente *OCaml*, é uma linguagem de programação funcional e fortemente tipada, da família ML com ênfase na expressividade e na segurança. É usada em aplicações sensíveis onde um único erro pode custar milhões. Será utilizada na implementação de nosso compilador.

Capítulo 2

Instalações

2.1 Python

Já vem instalado por padrão em sistemas *GNU/Linux*, para conferir a versão, digite no terminal:

```
> which python
```

2.2 Java

Para checar as versões disponíveis, digite no terminal:

```
>sudo dnf search openjdk
```

Instale a versão desejada digitando no terminal:

```
>sudo dnf install <openjdk-package-name>
```

Por exemplo:

```
>sudo dnf install java-1.8.0-openjdk.x86_64
```

Para verificar se foi instalado com sucesso digite:

```
>java -version
```

2.3 Dalvik

O Dalvik é uma VM executada em android, então neste trabalho usaremos o Android Studio. Para instalar basta ir no [site](#) baixar a versão que se aplica ao seu SO e configurar o PATH no terminal:


```
>export PATH=\$PATH:/diretoriolocal/android-studio/bin
```

Para executar o Android Studio, entre no diretório android-studio/bin e digite no terminal:

```
>sh studio.sh
```

Os componentes adicionais serão instalados com a ajuda do assistente de configuração na primeira execução do programa.

2.4 OCaml

Versão utilizada: 4.07.0

```
>sudo dnf install wget
>sudo dnf install git m4 mercurial darcs
>wget https://raw.githubusercontent.com/ocaml/opam/master/shell/
>opam_installer.sh -O - | sh -s /dev/bin
>opam init
>eval `opam config env`
>opam repository add git git+https://github.com/ocaml/
>opam-repository
>opam update
```

Para saber qual a versão mais atual:

```
>opam switch
```

Instalando a versão 4.0.7:

```
>opam switch 4.07.0
>eval `opam config env`
```

Instalar o rlwrap para trabalhar melhor com o OCaml:

```
> sudo dnf update
> sudo dnf install rlwrap
```

Executar o OCaml com o rlwrap:

```
>rlwraper ocaml
```

2.5 Compilação

Como gerar .smeli a partir do .dex.

2.5.1 Compilando Java em .dex

Dentro do diretório execute no terminal:

```
> javac file.java
> diretorioSDK/build-tools/version/dx --dex --output=file.dex file.class
```

Onde esta "diretorioSDK" é o caminho no qual foi instalado o SDK com o assistente de configuração do Android Studio, assim, como "version" a versão que esta sendo utilizada. Baixe o baksmali e smali, os dois na versão 2.2.6, no [site](#) para fazer o desassembly. É importante ressaltar que tem que deixar o .dex/.smali e o baksmali/smali no mesmo diretório.

2.5.2 Complilando .dex em .smali

Dentro do diretório execute no terminal:

```
> java -jar baksmali-2.2.6.jar disassemble file.dex
```

2.5.3 Compilando .smali code em .dex

Dentro do diretório execute no terminal:

```
> java -jar smali-2.2.6.jar assemble file.smali -o file.dex
```

2.6 Executando o arquivo .dex no Android

Depois de compilado o .dex, para executa-lo usaremos um emulador que rode o sistema operacional android. Usaremos o modelo Nexus 5, android 5.1.

2.6.1 Utilizando Emulador AVD

Após instalado o Android Studio, execute-o e crie um novo projeto em branco. Na interface do programa, vá em: *Tools > ADV Manager*.

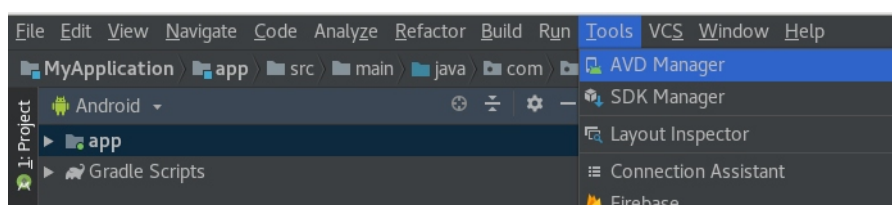


Figura 2.1: *ADV Manager*

A seguinte tela irá aparecer:

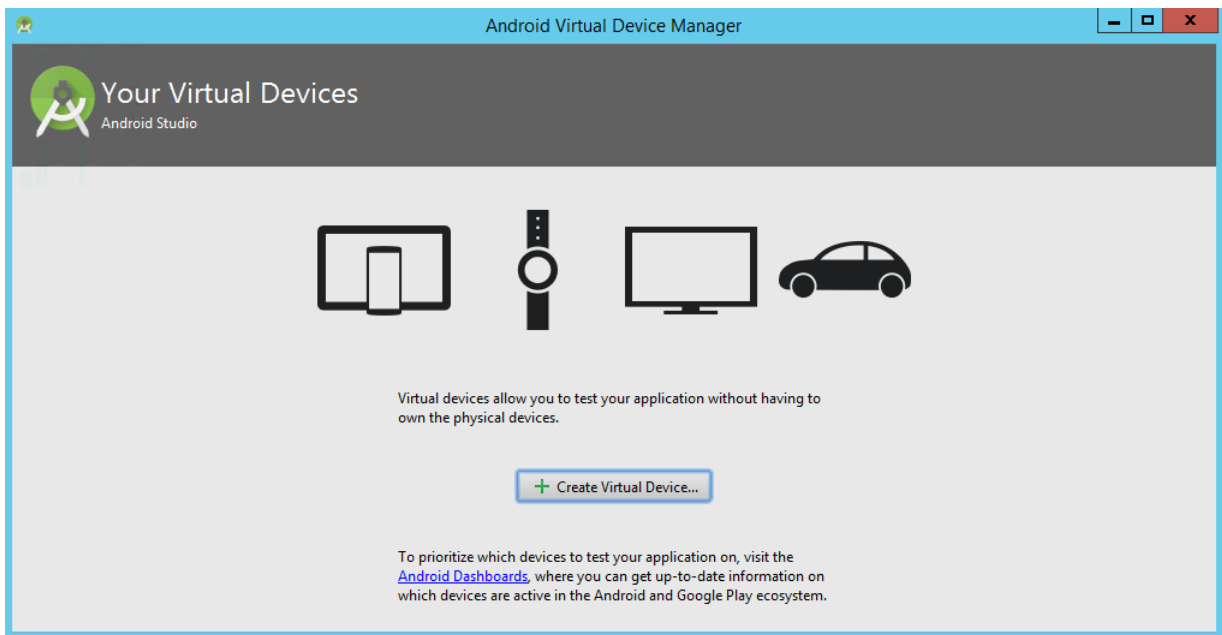


Figura 2.2: Criar um novo device

Nessa janela, selecione *Create Virtual Device*.

Logo em seguida irá aparecer essa janela:

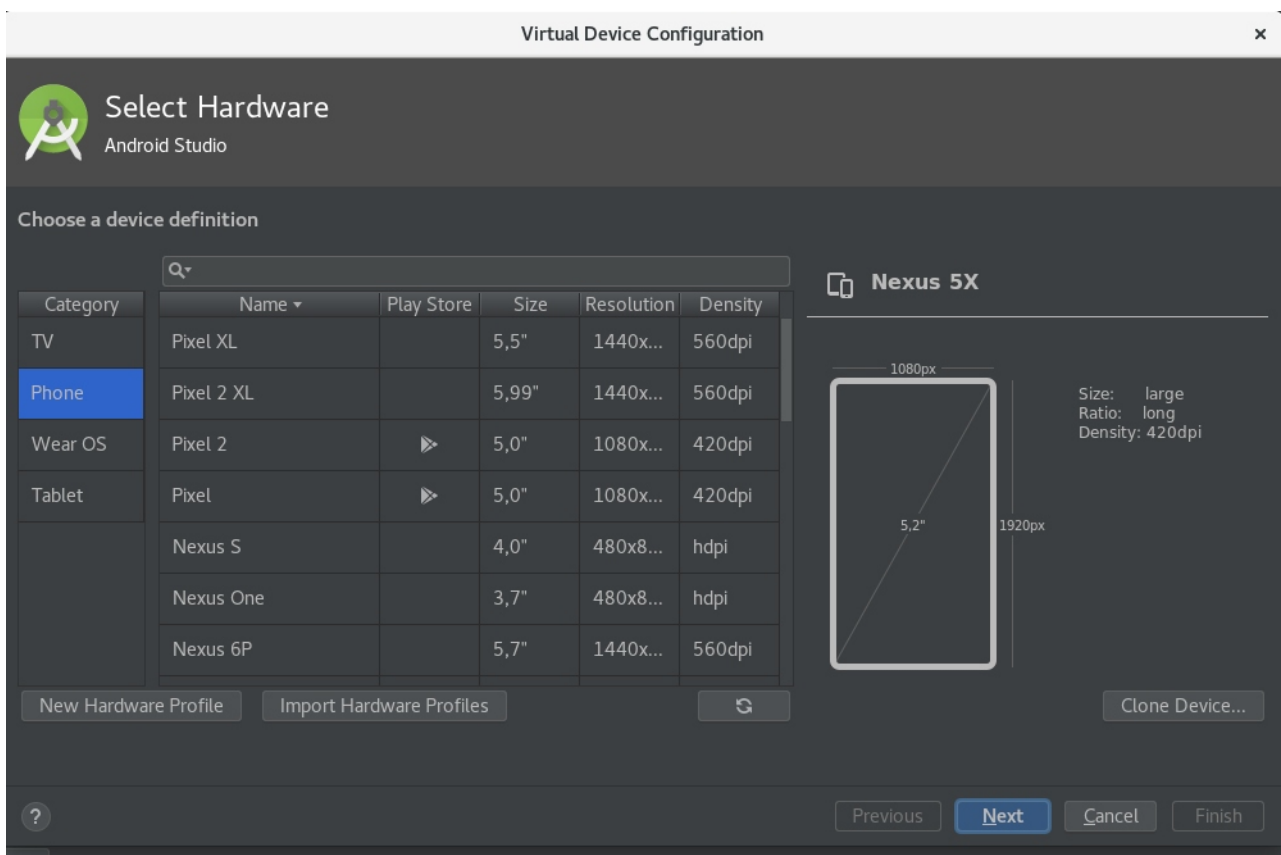


Figura 2.3: Modelo Device

Selecione o modelo que deseja emular, indico usar modelos Nexus, continue para as próximas

janelas clicando em next.

Abrindo o ADV Manager novamente, seu modelo emulado deverá aparecer como mostrado na figura abaixo, para iniciar, clique no ícone *Play*.

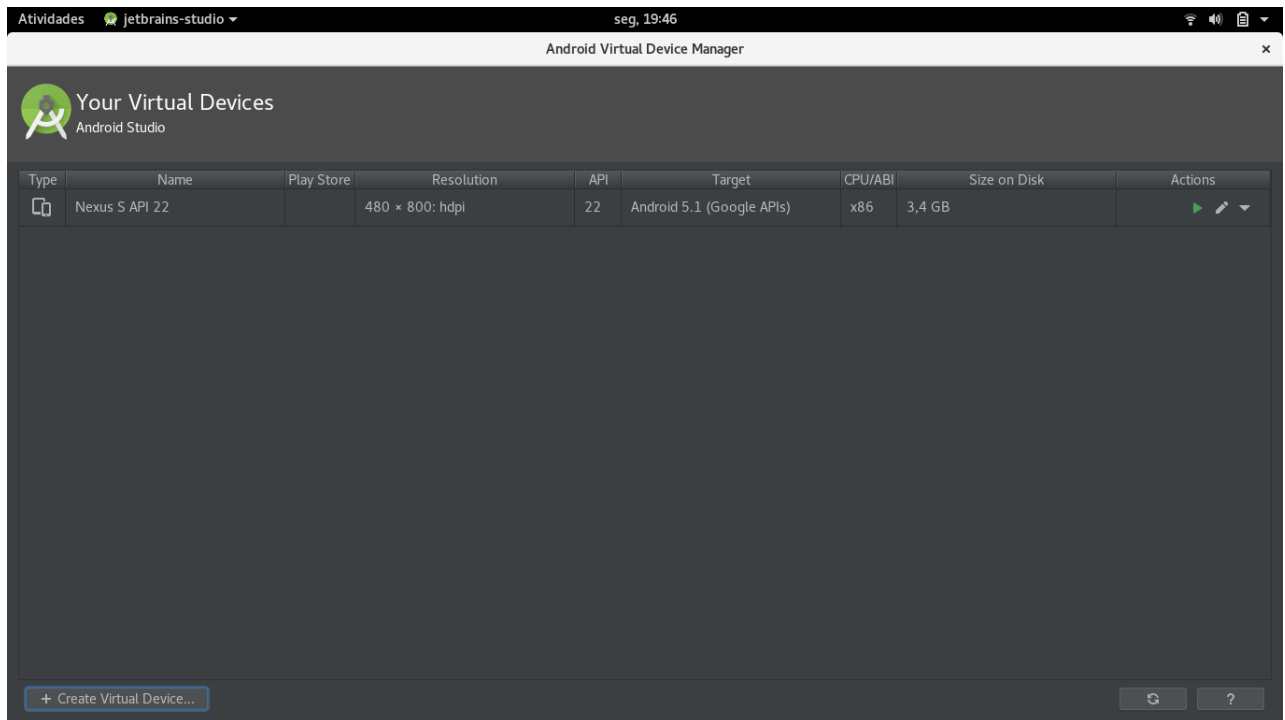


Figura 2.4: *Manager Devices*

Ao fazer esses passos irá aparecer um emulador da tela do celular, com isso já podemos rodar arquivos .dex.

No terminal digite:

```
> ./adb devices
> ./adb push /home/fenrir/dev/Projetos/compilados/nano01.dex /data/local
> ./adb shell dalvikvm -cp /data/local/nano01.dex nano01
```

A execução da ferramenta adb deve ser feita dentro do diretório *androidpath/sdk/platform-tools/*.

Capítulo 3

Nano Programas

Neste capítulo será apresentado alguns programas e suas respectivas versões em Python, Java e Smali Code.

3.1 Nano01

Listagem 3.1: Nano01.py

```
1 def main() -> None:
2     return
```

Listagem 3.2: Nano01.java

```
1 public class Nano01 {
2     public static void main(String[] args) {
3     }
4 }
```

Listagem 3.3: Nano01.smali

```
1 .class public LNano01;
2 .super Ljava/lang/Object;
3 .source "Nano01.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 1
19
```

```

20     .prologue
21     .line 3
22     return-void
23 .end method

```

Saída : nenhuma.

3.2 Nano02

Listagem 3.4: Nano02.py

```

1 def main() -> None:
2     n: int = 0

```

Listagem 3.5: Nano02.java

```

1 public class Nano02 {
2     public static void main(String[] args) {
3         int n;
4     }
5 }

```

Listagem 3.6: Nano02.smali

```

1 .class public LNano02;
2 .super Ljava/lang/Object;
3 .source "Nano02.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 1
19
20    .prologue
21    .line 4
22    return-void
23 .end method

```

Saída : nenhuma.

3.3 Nano03

Listagem 3.7: Nano03.py

```

1 def main() -> None:

```

3.4

```
2      n: int = 1
```

Listagem 3.8: Nano03.java

```
1 public class Nano03 {  
2     public static void main(String[] args) {  
3         int n;  
4         n = 1;  
5     }  
6 }
```

Listagem 3.9: Nano03.smali

```
1 .class public LNano03;  
2 .super Ljava/lang/Object;  
3 .source "Nano03.java"  
4  
5  
6 # direct methods  
7 .method public constructor <init>()V  
8     .registers 1  
9  
10    .prologue  
11    .line 1  
12    invoke-direct {p0}, Ljava/lang/Object;--><init>()V  
13  
14    return-void  
15 .end method  
16  
17 .method public static main([Ljava/lang/String;)V  
18     .registers 1  
19  
20     .prologue  
21     .line 4  
22     .line 5  
23     return-void  
24 .end method
```

Saída : nenhuma.

3.4 Nano04

Listagem 3.10: Nano04.py

```
1 def main() -> None:  
2     n: int = 1 + 2
```

Listagem 3.11: Nano04.java

```
1 public class Nano04 {  
2     public static void main(String[] args) {  
3         int n;  
4         n = 1+2;  
5     }  
6 }
```

Listagem 3.12: Nano04.smali

```

1 .class public LNano04;
2 .super Ljava/lang/Object;
3 .source "Nano04.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object; -> <init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 1
19
20    .prologue
21    .line 4
22    .line 5
23    return-void
24 .end method

```

Saída : nenhuma.

3.5 Nano05

Listagem 3.13: Nano05.py

```

1 def main() -> None:
2     n: int = 2
3     print(n)
4
5 main()

```

Listagem 3.14: Nano05.java

```

1 public class Nano05 {
2     public static void main(String[] args) {
3         int n;
4         n = 2;
5         System.out.println(n);
6     }
7 }

```

Listagem 3.15: Nano05.smali

```

1 .class public LNano05;
2 .super Ljava/lang/Object;
3 .source "Nano05.java"
4
5
6 # direct methods

```


3.6

```
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object;--><init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 3
19
20     .prologue
21     .line 4
22     const/4 v0, 0x2
23
24     .line 5
25     sget-object v1, Ljava/lang/System;-->out:Ljava/io/PrintStream;
26
27     invoke-virtual {v1, v0}, Ljava/io/PrintStream;-->println(I)V
28
29     .line 6
30     return-void
31 .end method
```

Saída : 2.

3.6 Nano06

Listagem 3.16: Nano06.py

```
1 def main() -> None:
2     n: int = 1 - 2
3     print(n)
4
5 main()
```

Listagem 3.17: Nano06.java

```
1 public class Nano06 {
2     public static void main(String[] args) {
3         int n;
4         n = 1-2;
5         System.out.println(n);
6     }
7 }
```

Listagem 3.18: Nano06.smali

```
1 .class public LNano06;
2 .super Ljava/lang/Object;
3 .source "Nano06.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
```

```

8      .registers 1
9
10     .prologue
11     .line 1
12     invoke-direct {p0}, Ljava/lang/Object; -><init>()V
13
14     return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 3
19
20     .prologue
21     .line 4
22     const/4 v0, -0x1
23
24     .line 5
25     sget-object v1, Ljava/lang/System; ->out:Ljava/io/PrintStream;
26
27     invoke-virtual {v1, v0}, Ljava/io/PrintStream; ->println(I)V
28
29     .line 6
30     return-void
31 .end method

```

Saída : -1.

3.7 Nano07

Listagem 3.19: Nano07.py

```

1 def main() -> None:
2     n = 1
3     if n == 1:
4         print(n)
5
6 main()

```

Listagem 3.20: Nano07.java

```

1 public class Nano07 {
2     public static void main(String[] args) {
3         int n;
4         n = 1;
5         if(n==1) {
6             System.out.println(n);
7         }
8     }
9 }

```

Listagem 3.21: Nano07.smali

```

1 .class public LNano07;
2 .super Ljava/lang/Object;
3 .source "Nano07.java"
4
5

```

```

6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10     .prologue
11     .line 1
12     invoke-direct {p0}, Ljava/lang/Object;--><init>()V
13
14     return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 3
19
20     .prologue
21     .line 4
22     const/4 v0, 0x1
23
24     .line 6
25     sget-object v1, Ljava/lang/System;-->out:Ljava/io/PrintStream;
26
27     invoke-virtual {v1, v0}, Ljava/io/PrintStream;-->println(I)V
28
29     .line 8
30     return-void
31 .end method

```

Saída : 1.

3.8 Nano08

Listagem 3.22: Nano08.py

```

1 def main() -> None:
2     n: int = 1
3     if n == 1:
4         print(n)
5     else:
6         print(0)
7
8 main()

```

Listagem 3.23: Nano08.java

```

1 public class Nano08 {
2     public static void main(String[] args) {
3         int n;
4         n = 1;
5         if(n==1){
6             System.out.println(n);
7         }else{
8             System.out.println(0);
9         }
10    }
11 }

```

Listagem 3.24: Nano08.smali

```

1 .class public LNano08;
2 .super Ljava/lang/Object;
3 .source "Nano08.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object; -> <init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 3
19
20     .prologue
21     .line 4
22     const/4 v0, 0x1
23
24     .line 6
25     sget-object v1, Ljava/lang/System; -> out:Ljava/io/PrintStream;
26
27     invoke-virtual {v1, v0}, Ljava/io/PrintStream; -> println(I)V
28
29     .line 10
30     return-void
31 .end method

```

Saída : 1.

3.9 Nano09

Listagem 3.25: Nano09.py

```

1 def main() -> None:
2     n: int = 1 + (1 / 2)
3     if n == 1:
4         print(n)
5     else:
6         print(0)
7
8 main()

```

Listagem 3.26: Nano09.java

```

1 public class Nano09 {
2     public static void main(String[] args) {
3         int n;
4
5         n = 1 + 1 / 2;
6         if(n==1) {

```

3.10

```
7         System.out.println(n);
8     } else {
9         System.out.println(0);
10    }
11 }
12 }
```

Listagem 3.27: Nano09.smali

```
1 .class public LNano09;
2 .super Ljava/lang/Object;
3 .source "Nano09.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 3
19
20     .prologue
21     .line 5
22     const/4 v0, 0x1
23
24     .line 7
25     sget-object v1, Ljava/lang/System;->out:Ljava/io/PrintStream;
26
27     invoke-virtual {v1, v0}, Ljava/io/PrintStream;->println(I)V
28
29     .line 11
30     return-void
31 .end method
```

Saída : 0.

3.10 Nano10

Listagem 3.28: Nano10.py

```
1 def main() -> None:
2     n: int = 1
3     m: int = 2
4     if n == m:
5         print(n)
6     else:
7         print(0)
8
9 main()
```

Listagem 3.29: Nano10.java

```

1 public class Nano10 {
2     public static void main(String[] args) {
3         int n, m;
4         n = 1;
5         m = 2;
6         if(n==m) {
7             System.out.println(n);
8         }else{
9             System.out.println(0);
10        }
11    }
12
13 }

```

Listagem 3.30: Nano10.smali

```

1 .class public LNano10;
2 .super Ljava/lang/Object;
3 .source "Nano10.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 3
19
20     .prologue
21     .line 4
22     .line 9
23     sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;
24
25     const/4 v1, 0x0
26
27     invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(I)V
28
29     .line 11
30     return-void
31 .end method

```

Saída : 0.

3.11 Nano11

Listagem 3.31: Nano11.py

```

1 def main() -> None:

```

```

2      n: int = 1
3      m: int = 2
4      x: int = 5
5      while x > n:
6          n = n + m
7          print(n)
8
9 main()

```

Listagem 3.32: Nano11.java

```

1 public class Nano11 {
2     public static void main(String[] args) {
3         int n, m, x;
4
5         n = 1;
6         m = 2;
7         x = 5;
8         while (x>n) {
9             n = n + m;
10            System.out.println(n);
11        }
12    }
13 }

```

Listagem 3.33: Nano11.smali

```

1 .class public LNano11;
2 .super Ljava/lang/Object;
3 .source "Nano11.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object;-><init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 5
19
20     .prologue
21     .line 5
22     const/4 v0, 0x1
23
24     .line 6
25     const/4 v1, 0x2
26
27     .line 7
28     const/4 v2, 0x5
29
30     .line 8
31     :goto_3
32     if-le v2, v0, :cond_c

```

```

33
34     .line 9
35     add-int/2addr v0, v1
36
37     .line 10
38     sget-object v3, Ljava/lang/System; ->out:Ljava/io/PrintStream;
39
40     invoke-virtual {v3, v0}, Ljava/io/PrintStream; ->println(I)V
41
42     goto :goto_3
43
44     .line 12
45     :cond_c
46     return-void
47 .end method

```

Saída : 3.

Saída : 5.

3.12 Nano12

Listagem 3.34: Nano12.py

```

1 def main() -> None:
2     n: int = 1
3     m: int = 2
4     x: int = 5
5     while x > n:
6         if n == m:
7             print(n)
8         else:
9             print(0)
10        x = x - 1
11
12 main()

```

Listagem 3.35: Nano12.java

```

1 public class Nano12 {
2     public static void main(String[] args) {
3         int n, m, x;
4         n = 1;
5         m = 2;
6         x = 5;
7         while(x>n) {
8             if(n==m)
9                 System.out.println(n);
10            else
11                System.out.println(0);
12            x = x - 1;
13        }
14    }
15 }

```

Listagem 3.36: Nano12.smali

```

1 .class public LNano12;

```



```

2 .super Ljava/lang/Object;
3 .source "Nano12.java"
4
5
6 # direct methods
7 .method public constructor <init>()V
8     .registers 1
9
10    .prologue
11    .line 1
12    invoke-direct {p0}, Ljava/lang/Object;--><init>()V
13
14    return-void
15 .end method
16
17 .method public static main([Ljava/lang/String;)V
18     .registers 5
19
20     .prologue
21     .line 4
22     const/4 v1, 0x1
23
24     .line 6
25     const/4 v0, 0x5
26
27     .line 7
28     :goto_2
29     if-le v0, v1, :cond_d
30
31     .line 11
32     sget-object v2, Ljava/lang/System;-->out:Ljava/io/PrintStream;
33
34     const/4 v3, 0x0
35
36     invoke-virtual {v2, v3}, Ljava/io/PrintStream;-->println(I)V
37
38     .line 12
39     add-int/lit8 v0, v0, -0x1
40
41     goto :goto_2
42
43     .line 14
44     :cond_d
45     return-void
46 .end method

```

Saída : 0.

Saída : 0.

Saída : 0.

Saída : 0.

Capítulo 4

Referências

Python

Dalvik

Smali

OCaml