JPCERT CC®

JEB Plugin 開発チュートリアル 第2回

－DEXファイルの構造を理解する－
JEB PluginからDEXファイルを扱う方法を修得する

一般社団法人JPCERTコーディネーションセンター

目次

- 第0回 JEBとは？
- 第1回 JEB Pluginとは
  - 1. JEB Pluginの使い方
  - 2. JEB Pluginの構造
  - 3. JEBのUIを利用するためのAPI
  - 4. ViewとSignature
- **第2回 DEXファイルの構造を理解する**
  - **1. DEXファイルの構造**
  - **2. jeb.api.dex**
  - **3. クロスリファレンス**
- 第3回 バイトコードについての理解
  - 1. CodeItem
- 第4回 JEB PluginからASTを扱う

1. DEXファイルの構造

DEXファイルの構造を知る

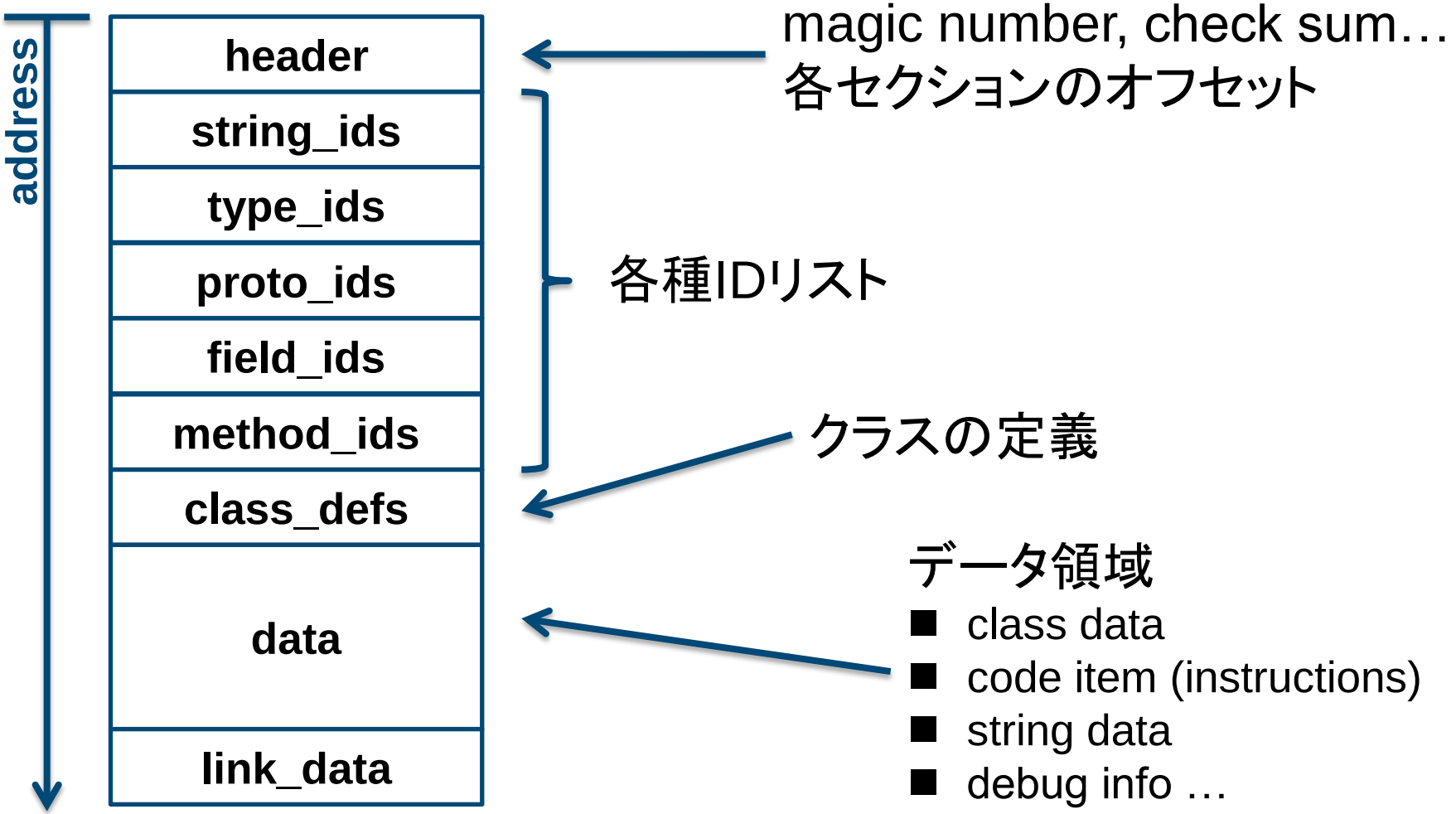
- Pluginを使ってDEXのバイトコードを操作する場合、ある程度DEXファイルに関する知識が必要

- 全てを覚える必要はない
- 概念的なことが分かっていればOK

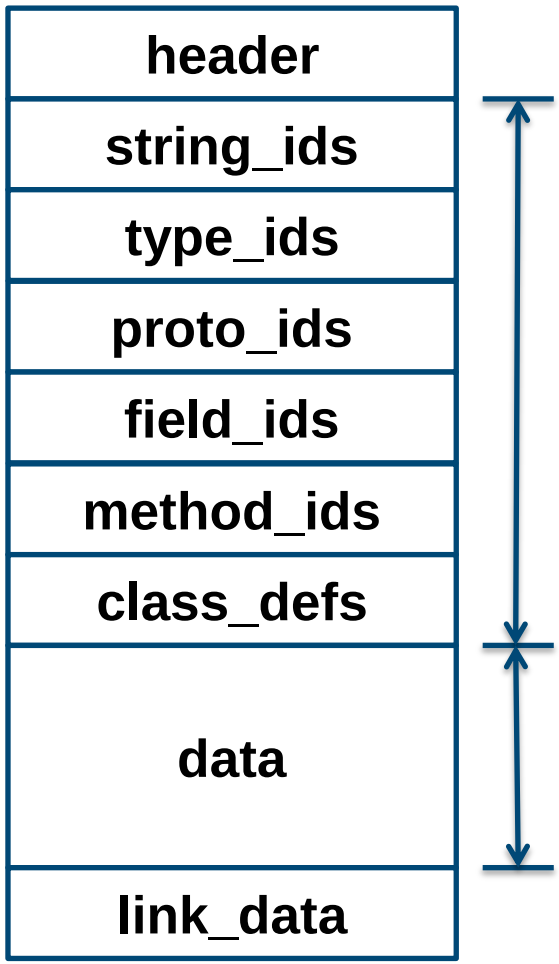
- 仕様
  - —http://source.android.com/devices/tech/dalvik/dex-format.html
- この回の資料は次のスライドを参考にしている
  - —http://www.slideshare.net/MasataNishida/dex-format-diff
  - —本資料への転載を承諾いただいた西田氏に感謝する

DEX File Section

DEXファイルの中身。次のような複数のセクションで構成されている

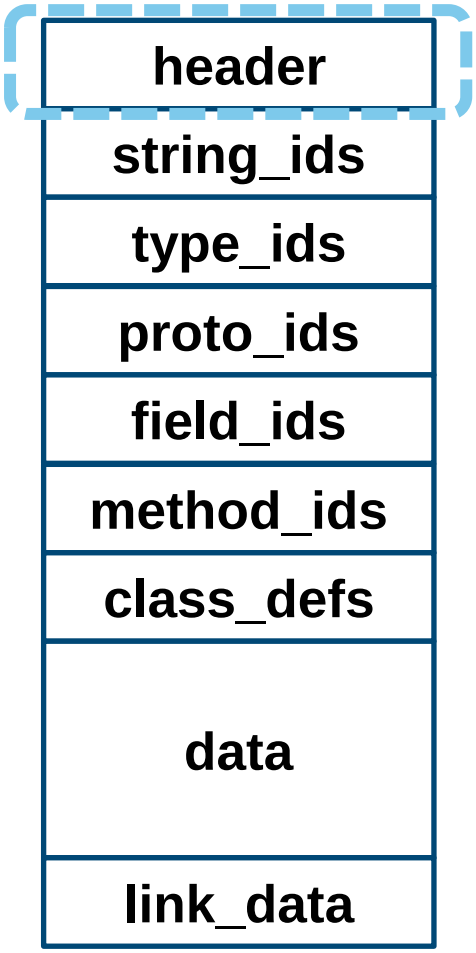


DEX File Section データの配置



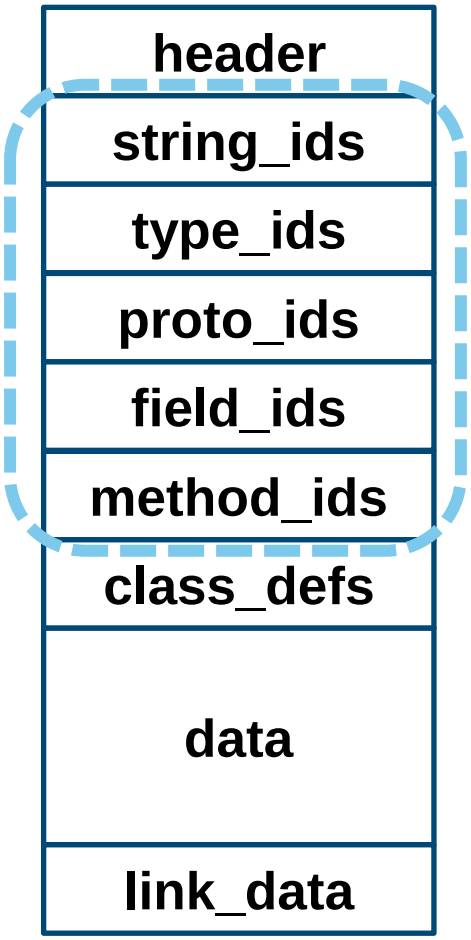
- 各種IDリスト、ClassDefs
  - 固定長データのリスト
  - オフセットと個数はヘッダに格納
- Dataセクション
  - 非固定長データが格納
  - 固定長データ内にDataセクションへのオフセットが記述される
  - Dataセクション内のデータからDataセクションを参照することも

DEX Header



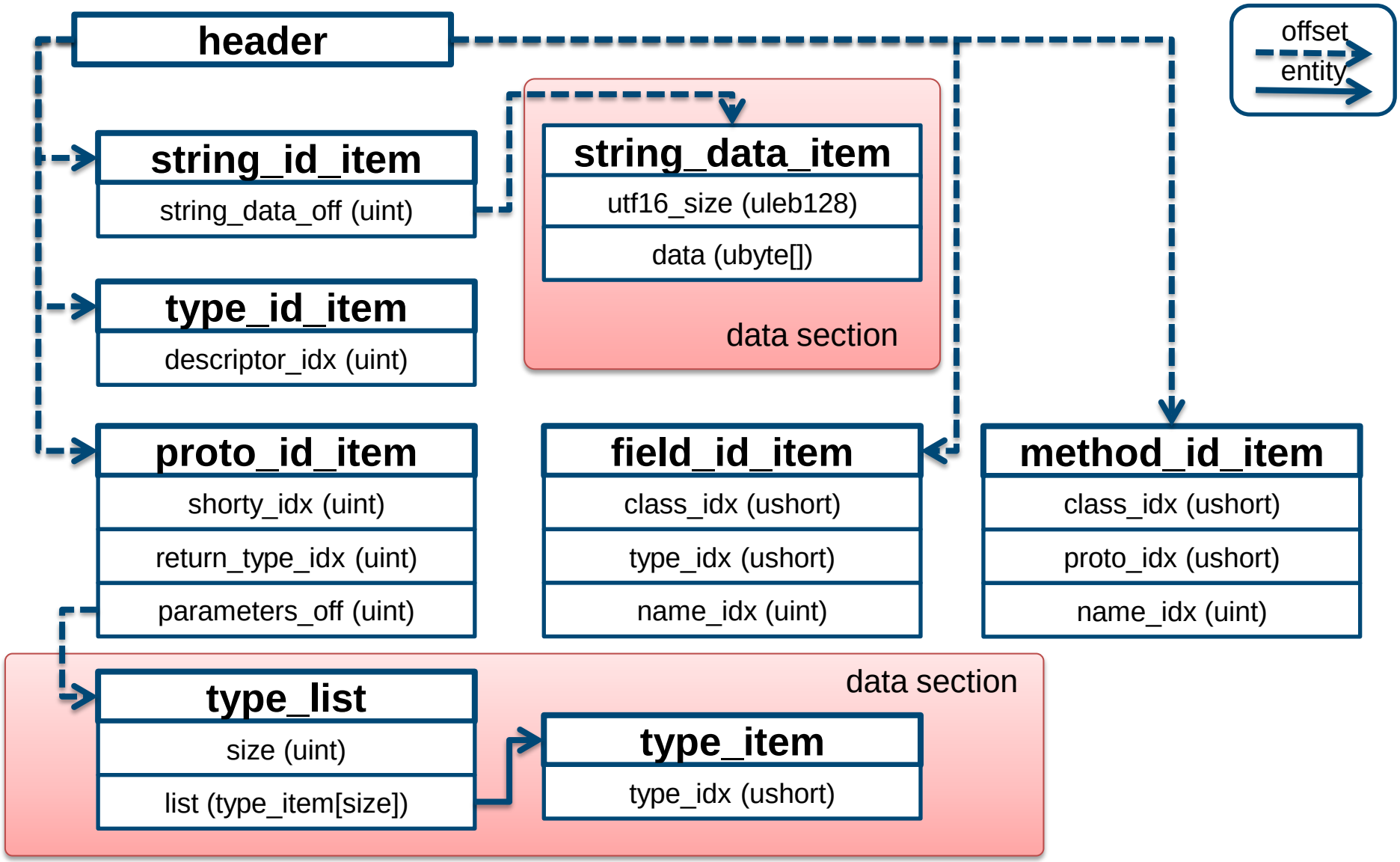
- magic: “dex¥a035¥0”
- checksum: Adler-32形式
- signature: SHA-1ハッシュ
- filesize
- …
- Section Sizes and Offsets
  - —string_ids_size, string_ids_off …

DEX IDリスト

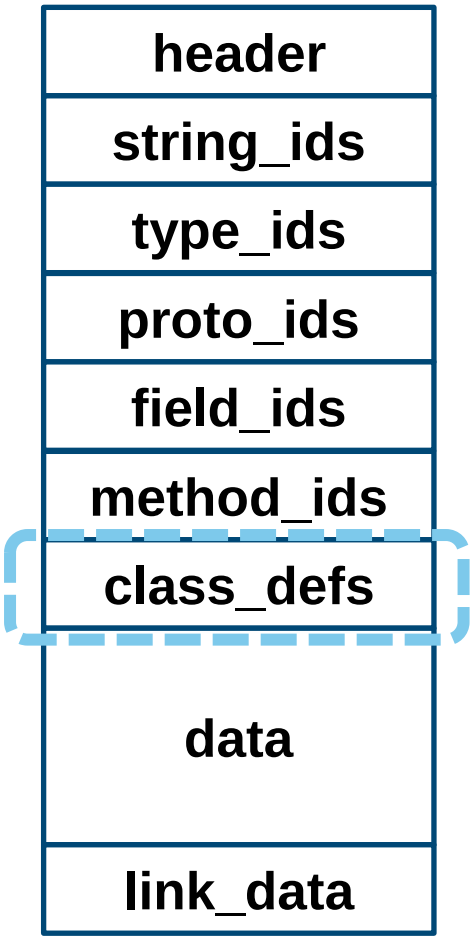


- string_ids
  - —文字列データへのoffset (data section)
- type_ids
  - —すべての型名の定義 (class, array, primitive)
  - —型名 → string_ids
- proto_ids
  - —method prototype identifiers list
  - —return type index, parameters offset
- field_ids
  - —class index, type index, name index
- method_ids
  - —class index, proto index, name index

headerと各IDリストの関係図

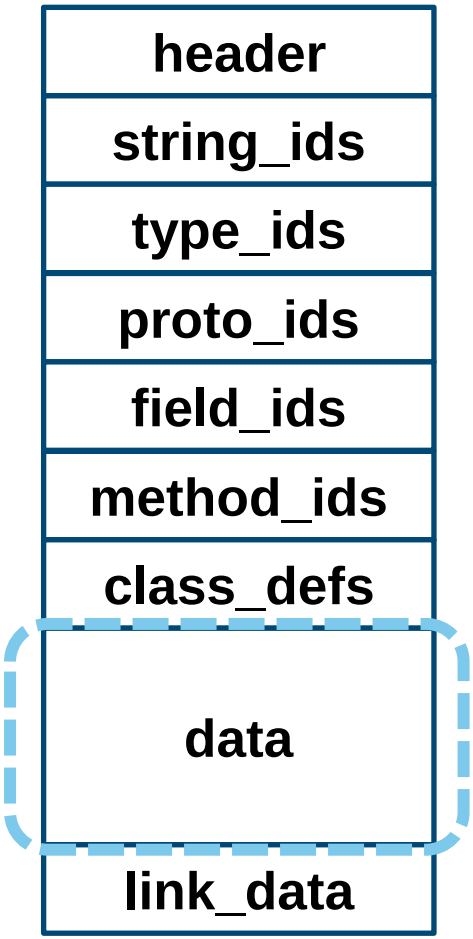


DEX Class Definitions



- class index
  - —type_idsのIndex
- access flags
  - —private, public, static, final …
- superclass index
  - —type_ids のIndex
- interfaces offset
  - —インターフェースリストへのオフセット
- class data offset
  - —クラスの内部的な情報へのoffset
  - —data sectionの中

DEX Data Section

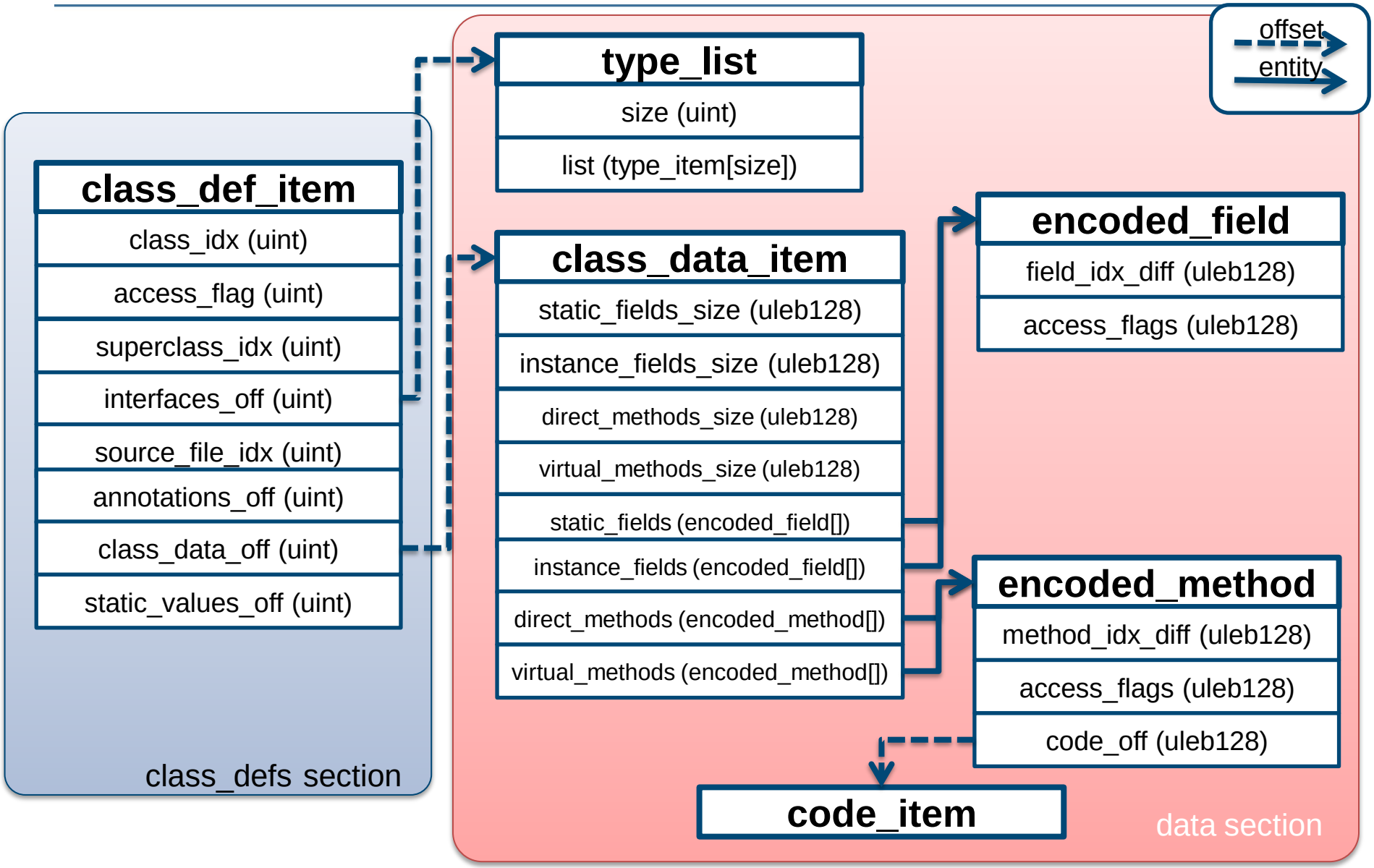


Data Sectionには非固定長のものが置かれる

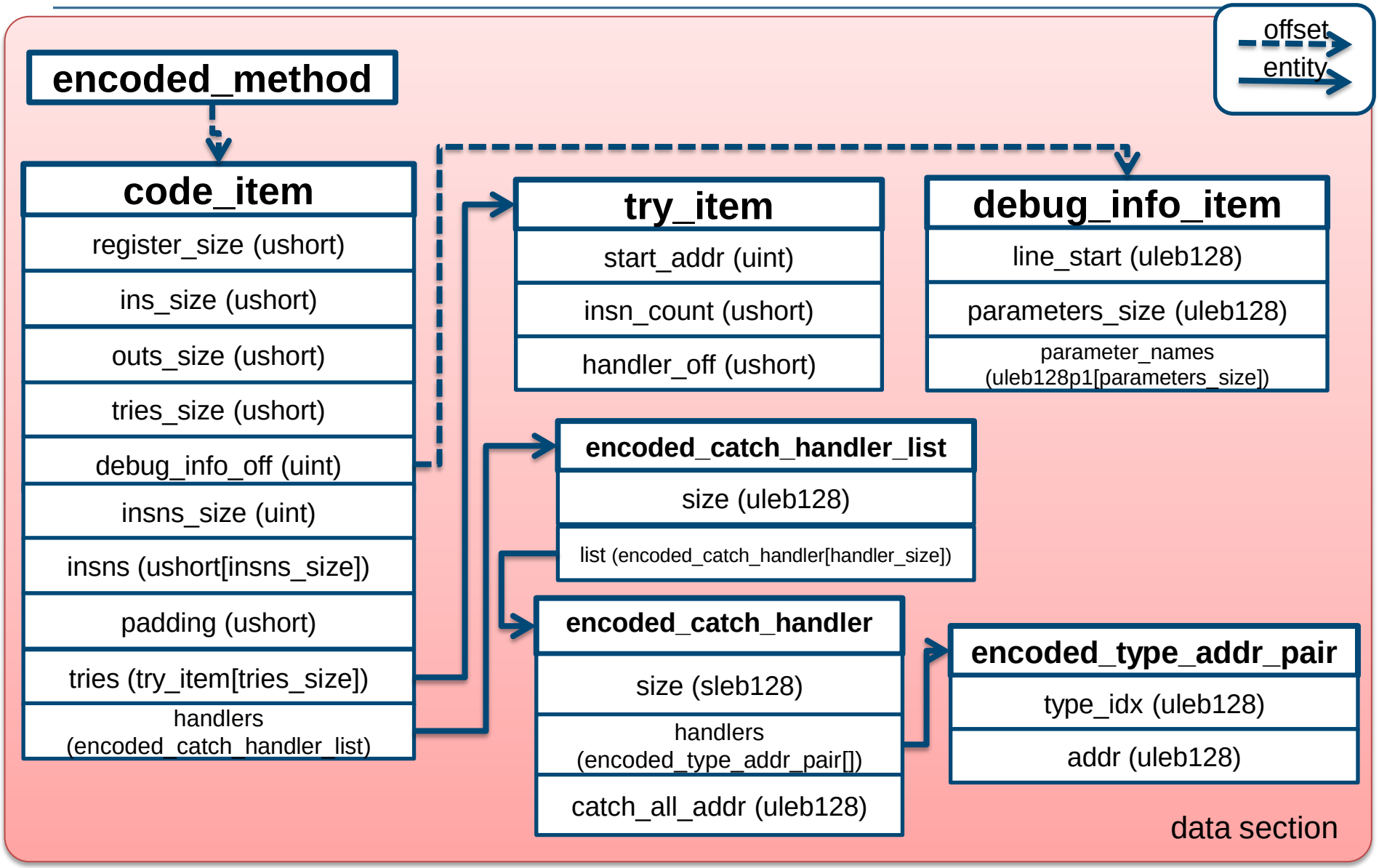
- class data item
  - —classのフィールド、メソッド情報
- code item
  - —メソッドごとに定義される
  - —instructions → 命令列
  - —try-catch handler → 例外処理
- string data item
  - —文字列データ (MUTF-8)
- debug info item

など

class_defsセクションとdataセクションの関係図



code_itemと各itemの関係図



2. jeb.api.dex

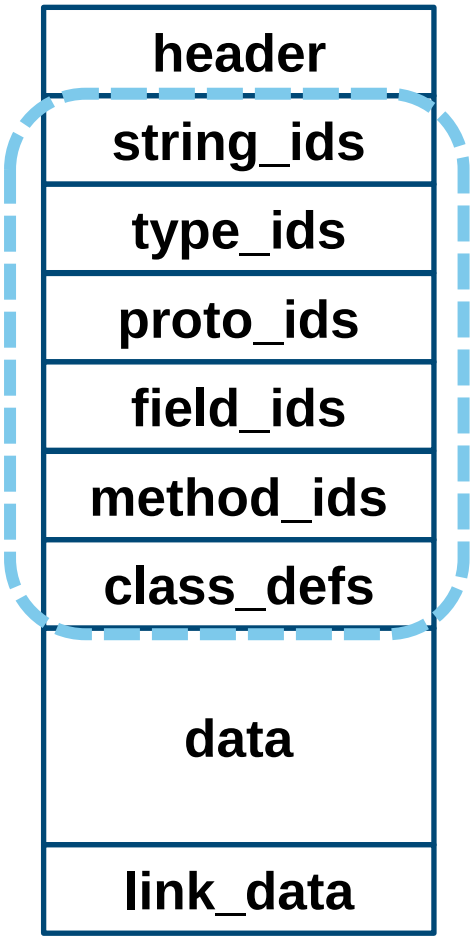
jeb.api.dex.Dex

- JEB PluginからDEXファイルを扱うAPI
  - —jeb.api.dex
- DEXファイルを表すクラス
  - —jeb.api.dex.Dex (Dexクラス)
    - クラス，フィールド，メソッドへのアクセス
    - JebInstance.getDex()メソッドで取得できる

Dexクラスのメソッドを使用することで、DEXファイルに含まれている各IDリストを取得することができる

- —取得したIDリストから、DEXファイルに含まれている文字列やクラス、メソッド、バイトコードといったものにアクセスできるようになる

IDリスト情報の取得



- IDリストとClass Defsに対応した情報の取得
- 例)Dex.getString(int **Index**)
  - —**Index**はIDリスト上の位置を表す
- リストの大きさはgetXxxCount()で取得

IDリストとの対応関係

ID List	method	return
string_ids	Dex.getString(index)	文字列
type_ids	Dex.getType(index)	文字列(型) (Stringに含まれている)
proto_ids	Dex.getPrototype(index)	DexPrototype メソッドのプロトタイプ (引数リストと返却値の型)
field_ids	Dex.getField(index)	DexField フィールド情報(型、名前)
method_ids	Dex.getMethod(index)	DexMethod
class_defs	Dex.getClass(index) Dex.getClass(name)	DexClass

各リストはソートされているが、一部例外有り
class_defs → スーパークラス、インターフェースが先に来るようになっている
(Class Loaderに起因するものか…?)

例題1

- DEXファイル内の文字列を表示しよう
  - —Dex.getString()とDex.getStringCount()の使い方を理解する

[例題1] DEXファイル内の文字列の表示

- 課題
  - DEXファイル内のすべての文字列をIndexと一緒に表示する
- 期待する出力結果

```
[0]
[1] :
[2] <
[3] <clinit>
[4] <init>
[5] >;
```

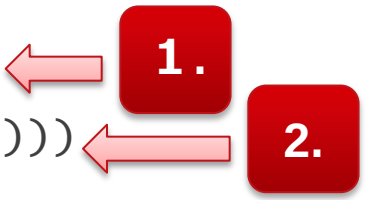
- ヒント
  - 文字列の取得
    - Dex.getString(idx)
  - 文字列テーブルの大きさの取得
    - Dex.getStringCount()
  - 出力フォーマット
    - `print("[%d] %s" % (idx, str))`

例題1の解答例

- StringList.py

[解説] DEXファイル内のStringの表示

```
from jeb.api import IScript

class StringList(IScript):

    def run(self, jeb):
        self.jeb = jeb
        self.dex = jeb.getDex()
        self.start()

    def start(self):
        for i in range(0, self.dex.getStringCount()):
            print("[%d] %s" % (i, self.dex.getString(i)))
```



1. 文字列テーブルの大きさを取得
2. getString()メソッドで文字列を取得し、出力する

クラス名の解決

- クラス、スーパークラス、interfaceなどは名前ではなくIndexで保持される
- Dex.getType(Index)でSignatureが返される

```
from jeb.api import IScript
from jeb.api.dex import Dex

class Example1(IScript):

    def run(self, jeb):
        dex = jeb.getDex()
        for cls_sig in dex.getClassSignatures(False):
            cls = dex.getClass(cls_sig)
            idx = cls.getClasstypeIndex()

            print "[%d] %s" % (idx, dex.getType(idx))
```

DexClassData, DexMethodData, DexFieldData

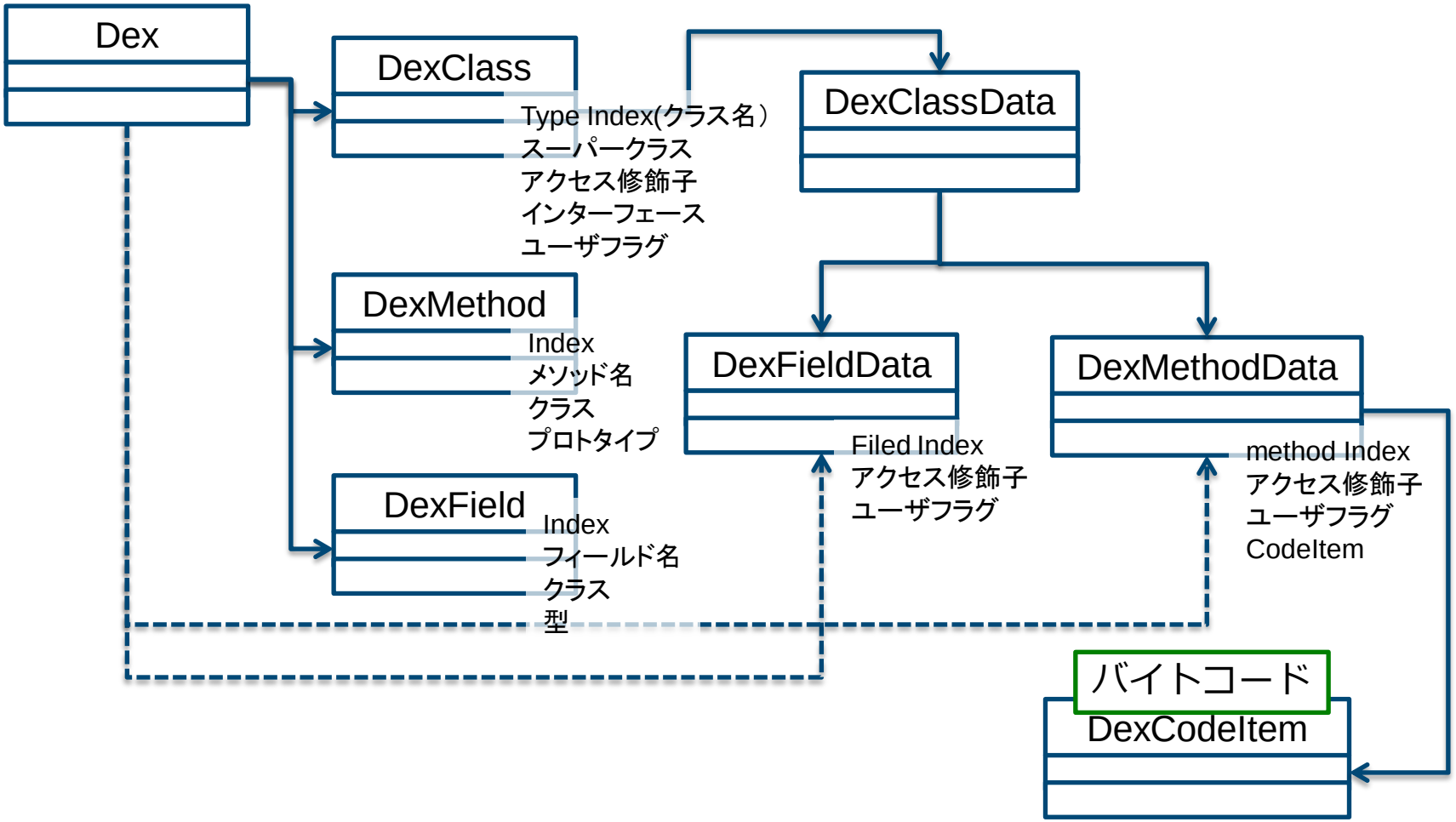
- JEBには、DEXファイルのデータセクション内に配置されるデータを表すクラスとして、次のクラスが用意されている

- DexClassData
  - —class_data_itemを表すクラス
- DexMethodData
  - —encoded_methodを表すクラス
- DexFieldData
  - —encoded_fieldを表すクラス

DexXXXDataクラスの取得方法

- DexClassDataを取得するには？
  - DexClass.getData()
    - DexClassData[]が返される
- DexMethodDataを取得するには？
  - DexClassData.getDirectMethods()
    - non-virtual(statics, constructors)のDexMethodData[]を取得できる
  - DexClassData.getVirtualMethods()
    - virtualメソッドのDexMethodData[]を取得できる
  - Dex.getMethodData(String name)
    - 引数にSignatureを渡すことでDexMethodData[]を取得できる
- DexFieldDataを取得するには？
  - DexClassData.getInstanceFields()
    - instanceフィールドのDexFieldData[]を取得できる
  - DexClassData.getStaticFields()
    - staticフィールドのDexFieldData[]を取得できる
  - Dex.getFieldData(String name)
    - 引数にSignatureを渡すことでDexFieldDataを取得できる

オブジェクトの取得ルートを図で表すと…



例題2

- DEXのクラス一覧を表示しよう
  - —dex.getClassCount()とdex.getClassSignature()の使い方を理解する

[例題2] DEXのクラス一覧の表示

- 課題
  - クラス名とそのスーパークラスのペアを標準出力に出力する
- 期待する出力結果

```
Lcom/example/sample01/BuildConfig; < Ljava/lang/Object;
Lcom/example/sample01/MainActivity; < Landroid/app/Activity;
Lcom/example/sample01/R$attr; < Ljava/lang/Object;
Lcom/example/sample01/R$dimen; < Ljava/lang/Object;
Lcom/example/sample01/R$drawable; < Ljava/lang/Object;
Lcom/example/sample01/R$id; < Ljava/lang/Object;
Lcom/example/sample01/R$layout; < Ljava/lang/Object;
Lcom/example/sample01/R$menu; < Ljava/lang/Object;
Lcom/example/sample01/R$string; < Ljava/lang/Object;
```

- ヒント
  - スーパークラスを取得するには
    - ClassData.getSuperclassIndex
    - dex.getType → class indexから文字列へ
  - dex.getClass()
    - 引数： Index(int) or Signature(string)
  - 一覧の取得
    - dex.getClassCount()でクラスの個数を取得する
      - dex.getClass(idex)
    - dex.getClassSignatures()でクラスのSignature一覧を取得

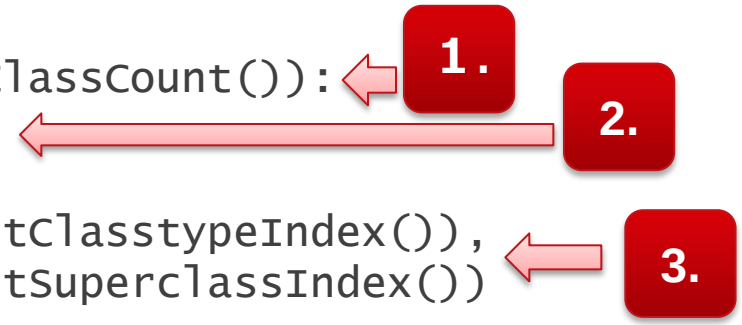
例題2の解答例

- ClassList.py

[解説] DEXのクラス一覧の表示 getClassCount()を使用する

- 二通りの方法が考えられる
  - getClassCount()を使用する方法
  - getClassSignatures()を使用する方法

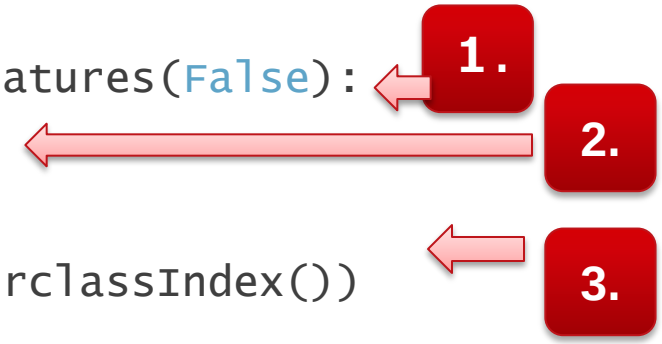
```
def list1(self, dex):
    for i in range(0, dex.getClassCount()):
        cls = dex.getClass(i)
        str = "%s < %s" % (
            dex.getType(cls.getClasstypeIndex()),
            dex.getType(cls.getSuperclassIndex())
        )
        print str
```



1. getClassCount()でクラス数を取得し、クラスの数だけループさせる
2. getClass()にIndexを渡してクラスを取得
3. getType()を使用してSignatureを取得する
   - DexClass.getClasstypeIndex()
     - クラスのIndexを取得する
   - DexClass.getSuperclassIndex()
     - スーパークラスのIndexを取得する

[解説] DEXのクラス一覧の表示 getClassSignatures()を使用する

```
def list2(self, dex):
    for cls_sig in dex.getClassSignatures(False):
        cls = dex.getClass(cls_sig)
        str = "%s < %s" % (
            cls_sig,
            dex.getType(cls.getSuperclassIndex())
        )
        print str
```



1. getClassSignatures()を使用してSignatureを取得
2. getClass()にSignatureを渡してクラスを取得
3. getType() にスーパークラスのIndex渡して、スーパークラスのSignatureを取得

例題3

- Assembly Viewでフォーカスしているクラスのメソッド一覧の表示
  - —DexClassData.getVirtualMethods()とDexClassData.getDirectMethods()の使い方を理解する

[例題3] Assembly Viewでフォーカスしているクラスのメソッド一覧の表示

- 課題
  - Assemblyでフォーカスしているクラスのメソッド一覧を取得する
- 期待する出力結果

```
target class: Lcom/example/sample01/MainActivity;
direct methods:
    Lcom/example/sample01/MainActivity;-><init>
virtual methods:
    Lcom/example/sample01/MainActivity;->onCreate
    Lcom/example/sample01/MainActivity;->onCreateOptionsMenu
```

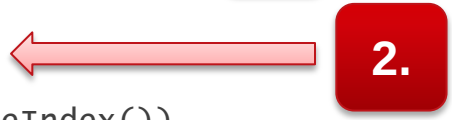
- ヒント
  - CodePositionからSignatureを取得
    - SignatureはClass,Method,Fieldである可能性
      - '->'以降を削除してクラス名に変換
  - クラス名からDexClass → DexClassDataを取得
    - DexClassData.getVirtualMethods, getDirectMethodsでメソッドの一覧を取得
  - それぞれの名前を表示する

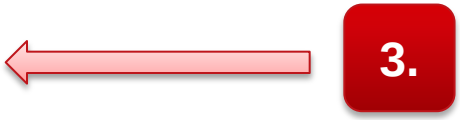
例題3の解答例

- FocusedMethodList.py

[解説] Assembly Viewでフォーカスしているクラスのメソッド一覧の表示

```
def start(self):
    # get assembly view
    view = self.ui.getView(View.Type.ASSEMBLY)
    # get signature from caret position
    msig = view.getCodePosition().getSignature()

    cls_sig = re.sub(r'->.*', '', msig)
    cls = self.dex.getClass(cls_sig)
    cls_name = self.dex.getType(cls.getClasstypeIndex())
    print 'target class: ' + cls_name

    cls_data = cls.getData()

    print 'direct methods:'
    for md in cls_data.getDirectMethods():
        meth = self.dex.getMethod(md.getMethodIndex())
        print "¥t%s->%s" % (cls_name, meth.getName())

    print 'virtual methods:'
    for md in cls_data.getVirtualMethods():
        meth = self.dex.getMethod(md.getMethodIndex())
        print "¥t%s->%s" % (cls_name, meth.getName())
```





1. キャレット位置からSignatureを取得
2. Signatureからメソッド名を削除してクラスのSignatureを作り、クラスを取得
3. 取得したクラスからgetData()を使用してクラスデータを取得
4. getDirectMethods()やgetVirtualMethods()を使用して、Direct MethodやVirtual Methodを取得する

例題4

- 難読化対策ヘルパーPluginの作成
  —Jeb.renameClass()の使い方を理解する

[例題4] 難読化対策ヘルパーPluginの作成

- 課題
  - クラス名に”_Activity”など、スーパークラスのクラス名を付加するPluginを作成する
- 期待する出力結果

```
rename from 'Lcom/example/sample01/a;' to 'Lcom/example/sample01/a_Activity;'
```

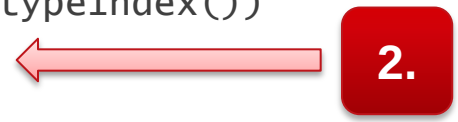
- ヒント
  - DexClassの一覧を取得する
  - スーパークラスが対象の場合リネームする
  - スーパークラスの取得
    - DexClass.getSuperclassIndex()
  - リネームを行うメソッドは？
    - Jeb.renameClass(src, dest)
      - src: Lxxx/yyy/zzz/ClsName;の形式
      - dest: パッケージ名のないクラス名 → ClsName
  - 対象：スーパークラスが以下のものの場合クラス名を変更する
    - Landroid/app/Activity;
    - Landroid/app/Service;
    - Landroid/app/BroadcastReceiver;

例題4の解答例

- RenameClasses.py

[解説] 難読化対策ヘルパーPluginの作成 (rename_classes)

```
def rename_classes(self):
    for i in range(0, self.dex.getClassCount()):
        cls = self.dex.getClass(i)
        cls_name = self.dex.getType(cls.getClasstypeIndex())
        ret = self.rename_by_super_class(cls)

        if ret:
            new_name = self.dex.getType(cls.getClasstypeIndex())
            print "rename from '%s' to '%s'" % (cls_name, new_name)
```

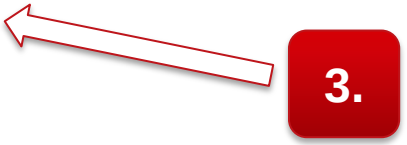


1. クラスの一覧を取得
2. 取得したクラスをrename_by_super_class関数に渡す
   - rename_by_super_class関数ではリネーム対象かどうかチェックしている (次頁参照)
3. リネームした結果を出力

[解説] 難読化対策ヘルパーPluginの作成 (rename_by_super_class)

- リネーム対象かどうかをチェックする関数 (rename_by_super_class)

```
def rename_by_super_class(self, cls):
    rename_targets = {
        'Landroid/app/Service;': 'Service',
        'Landroid/app/Activity;': 'Activity',
        'Landroid/content/BroadcastReceiver;': 'Receiver',
        'Landroid/content/ContentProvider;': 'Provider',
        'Ljava/lang/Thread;': 'Thread',
        'Landroid/os/AsyncTask;': 'AsyncTask',
    }
    super_cls_type = self.dex.getType(cls.getSuperclassIndex())
    if super_cls_type in rename_targets.keys():
        val = rename_targets[super_cls_type]
        return self.append_cls_name(cls, val)
    else:
        return False
```



1. リネーム対象を設定する
2. 渡されたクラスがリネーム対象かどうかチェックする
3. rename_targets関数でリネームする

[解説] 難読化対策ヘルパーPluginの作成 (append_cls_name)

- リネームを行う関数 (append_cls_name)

```
def append_cls_name(self, cls, append_str):
    p = re.compile("^.*¥/([¥w$]+);$")
    # cls_name has package path
    cls_name = self.dex.getType(cls.getClasstypeIndex())
    if cls_name.find(append_str) == -1:
        s = re.search(p, cls_name)
        simple_new_name = s.group(1) + '_' + append_str
        return self.jeb.renameClass(cls_name, simple_new_name)
    else:
        return False
```

1. 正規表現オブジェクトを作る
2. Signatureに追加する文字列が含まれている場合、Signatureの末尾にその文字列を追加する
3. renameClass()メソッドを使用してクラスをリネームする

3. クロスリファレンス

クロスリファレンス

- JEBのクロスリファレンス
  - 要素を参照している箇所(メソッド)の一覧を取得できる
- Pluginからクロスリファレンスの機能が使用できる
  - Dex.getFiledReferences(index)
    - フィールドのリファレンス
  - Dex.getMethodReferences(index)
    - メソッドのリファレンス
  - Dex.getTypeReferences(index)
    - タイプのリファレンス
- 返り値
  - メソッドIndexのリスト
  - 参照箇所がない場合はNone

Dex.getMethodSignatures() / Dex.getFieldSignatures()

- DEX内のメソッドまたフィールドのSignature一覧を取得するメソッド
  - Dex.getMethodSignatures()
    - メソッド Signatureの一覧を取得
  - Dex.getFieldSignatures()
    - フィールド Signatureの一覧を取得
  - 引数はTrue/False
    - True
      - JEBで変更した内容を反映したSignatureを取得
    - False
      - 変更していないSignatureを取得
- Dex.getMethod(Field)Signatures()を使うとIndexとSignatureを相互変換できる
  - Index から Signature を取得
    - Dex.getMethodSignatures(False)[index]
  - Signature から Indexを取得
    - Dex.getMethodSignatures(False).indexOf(Signature)

例題5

- 参照箇所一覧を表示しよう
  —Dex.getXxxxReferences()の使い方を理解する

[例題5] 参照箇所一覧を表示する

- 課題
  - フォーカスしているメソッドを参照している箇所の一覧を表示する
- 期待する出力結果

```
Lcom/example/contentprovider/MainActivity;->showListView()V(method)
    Lcom/example/contentprovider/MainActivity;->access$1(Lcom/example/contentprovider/MainActivity;)V
```

- ヒント
  - フォーカス位置のSignatureを取得
    - View.getCodePosition.getSignature()
  - Signatureの判定
    - クラス or メソッド or フィールド
      - “()”があるか，”->”があるかで判定
  - Dex.getXxxxReferences()で参照メソッドのIndexを取得
  - Dex.getMethodSignatures()でメソッド一覧の配列を取得
    - 参照メソッドのIndexを使用して、メソッド名を取得・表示する

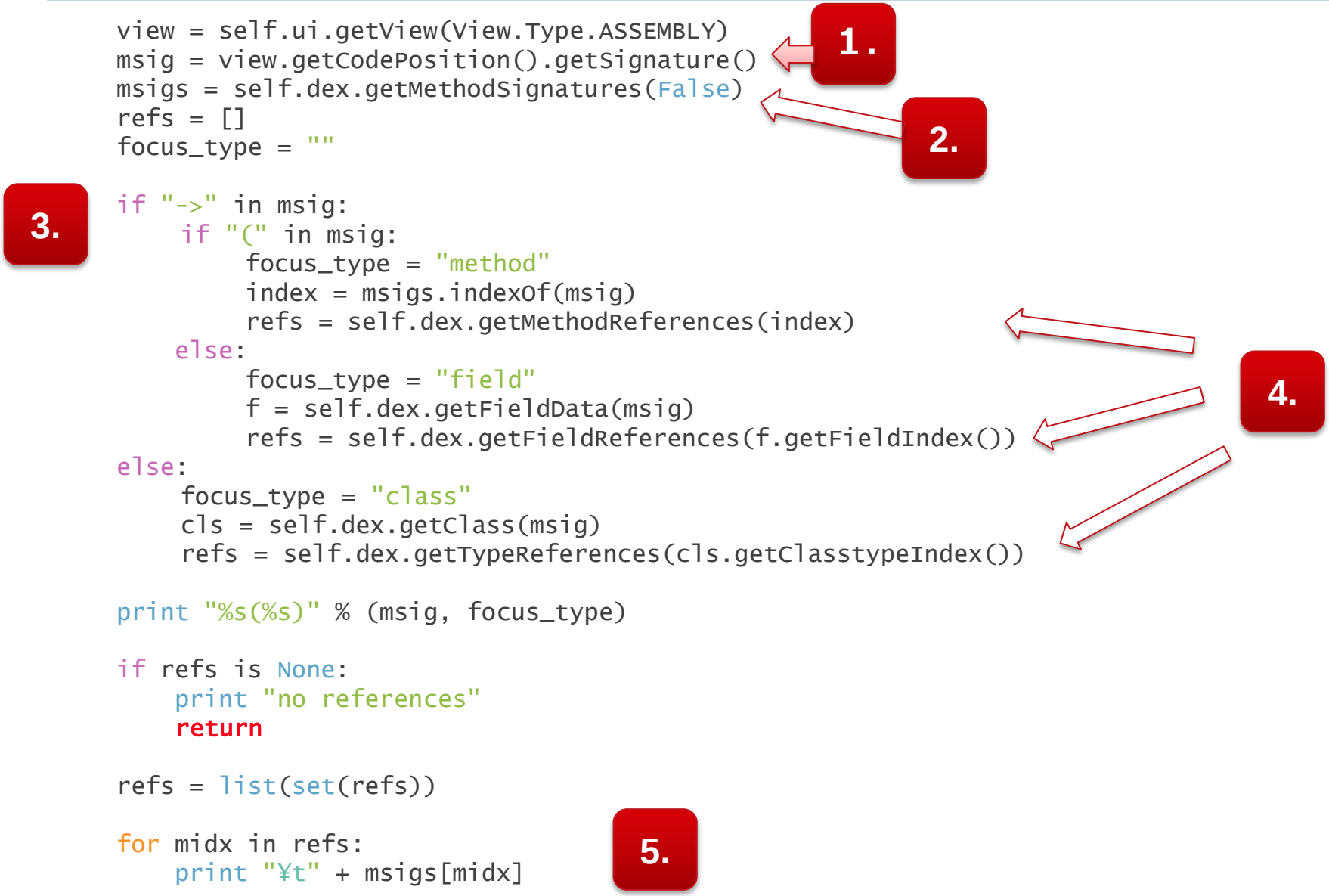
※ Signature配列の添字 = メソッドIndex

例題5の解答例

- References.py

[解説] 参照箇所一覧を表示する

```
view = self.ui.getView(View.Type.ASSEMBLY)
msig = view.getCodePosition().getSignature()
msigs = self.dex.getMethodSignatures(False)
refs = []
focus_type = ""

if "->" in msig:
    if "(" in msig:
        focus_type = "method"
        index = msigs.indexOf(msig)
        refs = self.dex.getMethodReferences(index)
    else:
        focus_type = "field"
        f = self.dex.getFieldData(msig)
        refs = self.dex.getFieldReferences(f.getFieldIndex())
else:
    focus_type = "class"
    cls = self.dex.getClass(msig)
    refs = self.dex.getTypeReferences(cls.getClasstypeIndex())

print "%s(%s)" % (msig, focus_type)

if refs is None:
    print "no references"
    return

refs = list(set(refs))

for midx in refs:
    print "¥t" + msigs[midx]
```



[解説] 参照箇所一覧を表示する

1. フォーカス位置のSignatureを取得する

```
msig = view.getCodePosition().getSignature()
```

1. メソッドSignatureの一覧を取得する

```
msigs = self.dex.getMethodSignatures(False)
```

1. 取得したSignatureがクラスかフィールドかメソッドかを判定する
   - Signatureに「->」が含まれていない場合
     - **クラス**
   - 「->」が含まれている場合
     - 「(」が含まれていない場合
       - **フィールド**
     - 「(」が含まれている場合
       - **メソッド**
2. 判定後、それぞれでDex.getXxxxReferences()で参照メソッドのIndexを持つ配列を取得する
3. 2.で取得したSignature一覧を元に、4.のIndexを持つSignatureを出力する