OPEN
WORLD
FORUM

THINKCODE
EXPERIMENT

OPEN WORLD FORUM | THINKCODE EXPERIMENT

Dalvik 292

Jérôme Pilliet

Université Paris-Est de Marne-la-Vallée

OPEN WORLD FORUM

Sommaire

- Avant propos
- JVM vs DVM
  - Principe
  - Comportement
- JSR 292
  - Principe
  - java.lang.invoke
  - Exemple
- Optimisation
  - VMs modernes
  - JSR 292
- Dalvik 292
  - Problèmes et solutions partielles
- Conclusion

Avant propos

- Langages typés dynamiquement
  - sémantique déterminée à l'exécution
- Smartphone / tablette
  - mémoire restreinte
  - puissance de calcul restreinte
- Android et Dalvik
  - OS le plus utilisé
  - technologie Java - Dalvik

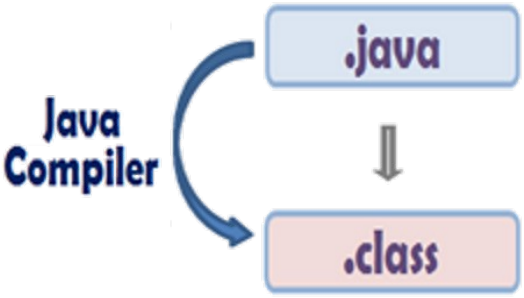
python™

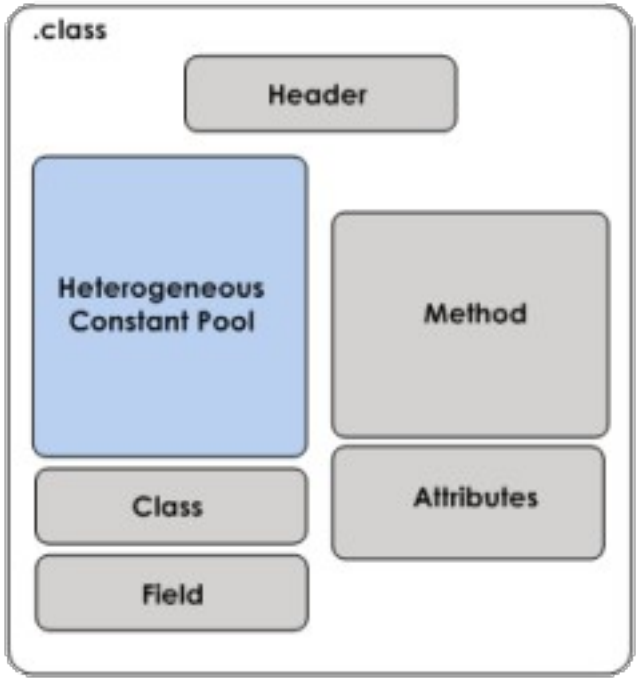
Ruby

Groovy

JVM vs DVM
Principe

- Machine à piles / machine à registres
- Spécification ~ Java 6

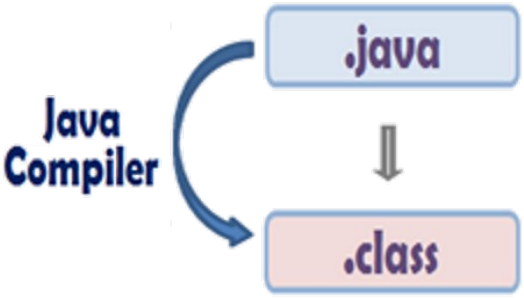


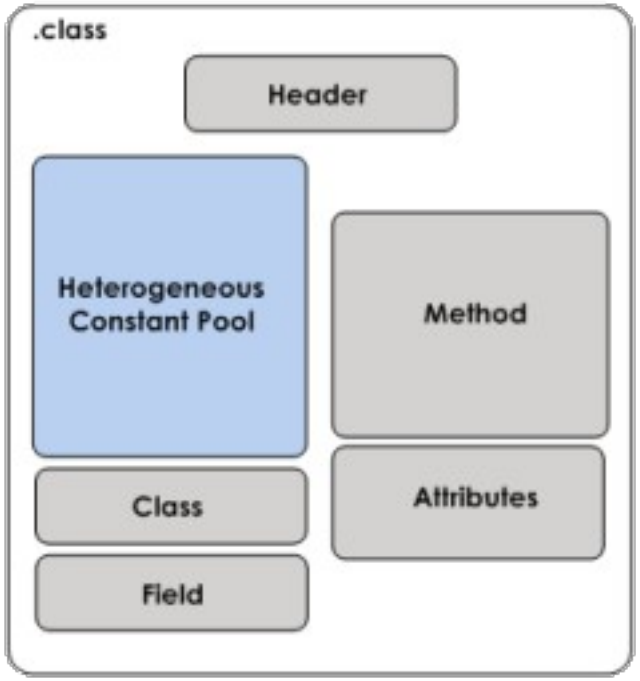


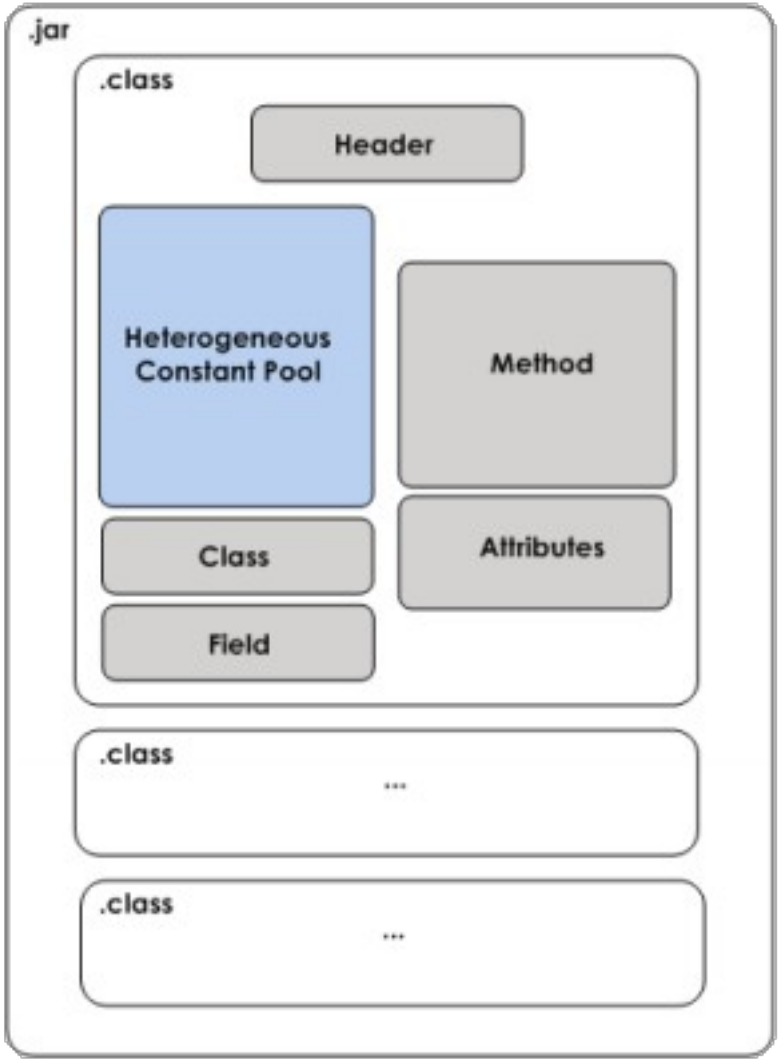
OPEN WORLD FORUM

JVM vs DVM
Principe

- Machine à piles / machine à registres
- Spécification ~ Java 6



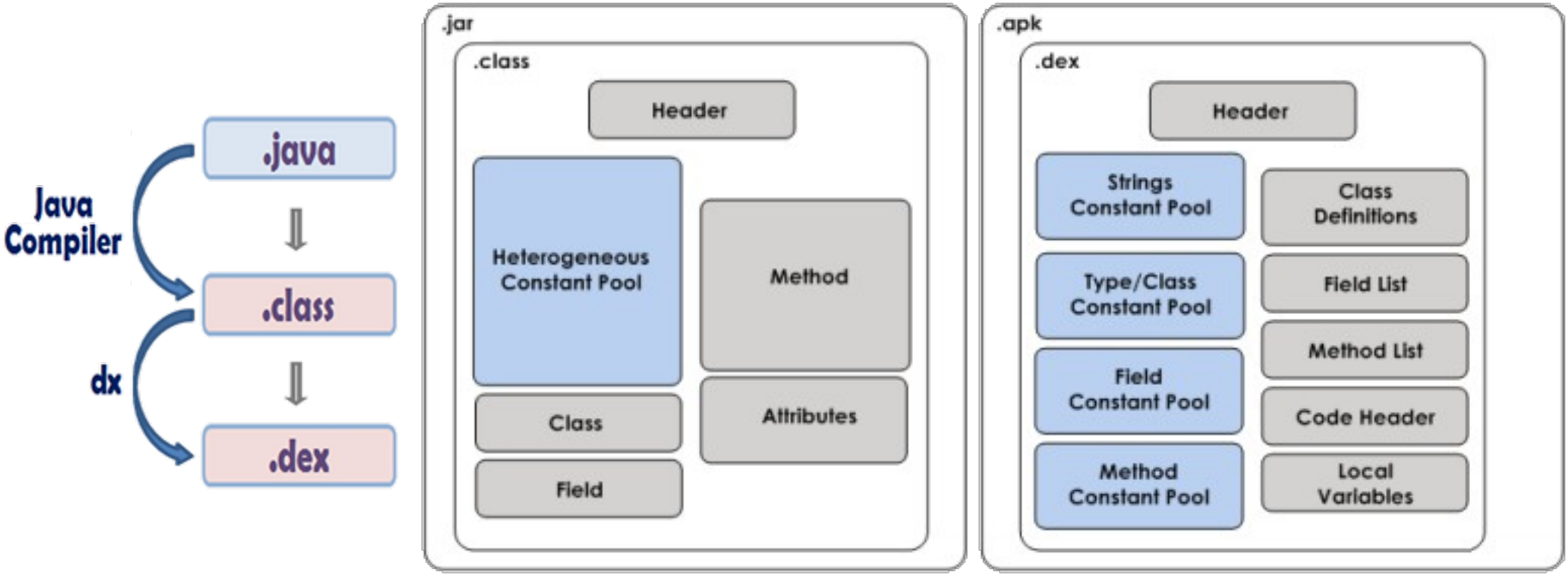




OPEN WORLD FORUM

JVM vs DVM
Principe

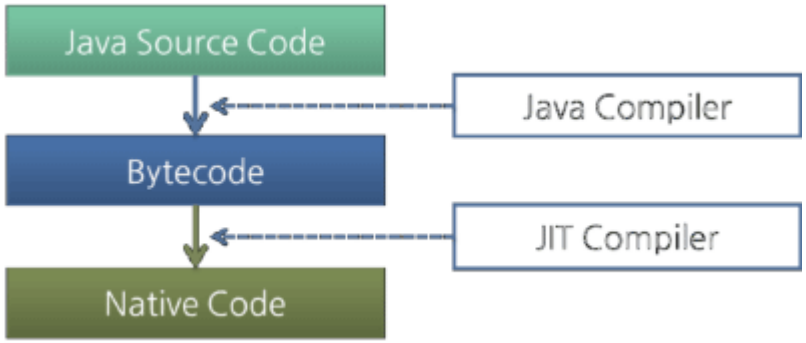
- Machine à piles / machine à registres
- Spécification ~ Java 6



JVM vs DVM
Comportement

- Dalvik
  - Bytecode
    - readonly
  - ODEX
    - bytecode
    - vtables (methodes)
    - offsets (attributs)
    - method inlining
  - Zygote
    - pre-load VM
  - Trace/Method JIT Compiler
  - Garbage Collector
    - restreint

- Java Virtual Machine
  - JIT Compiler

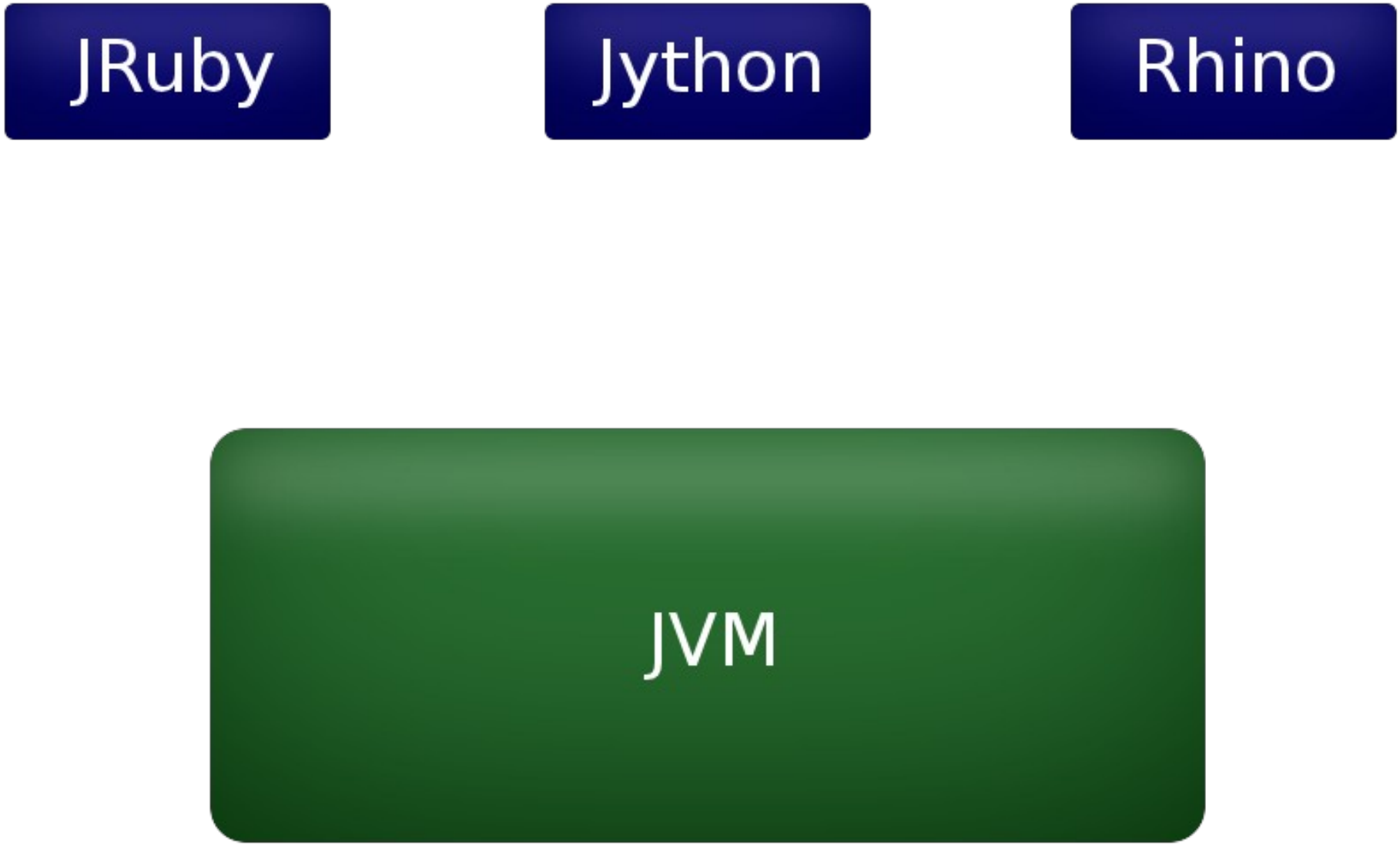


JSR 292
Principe

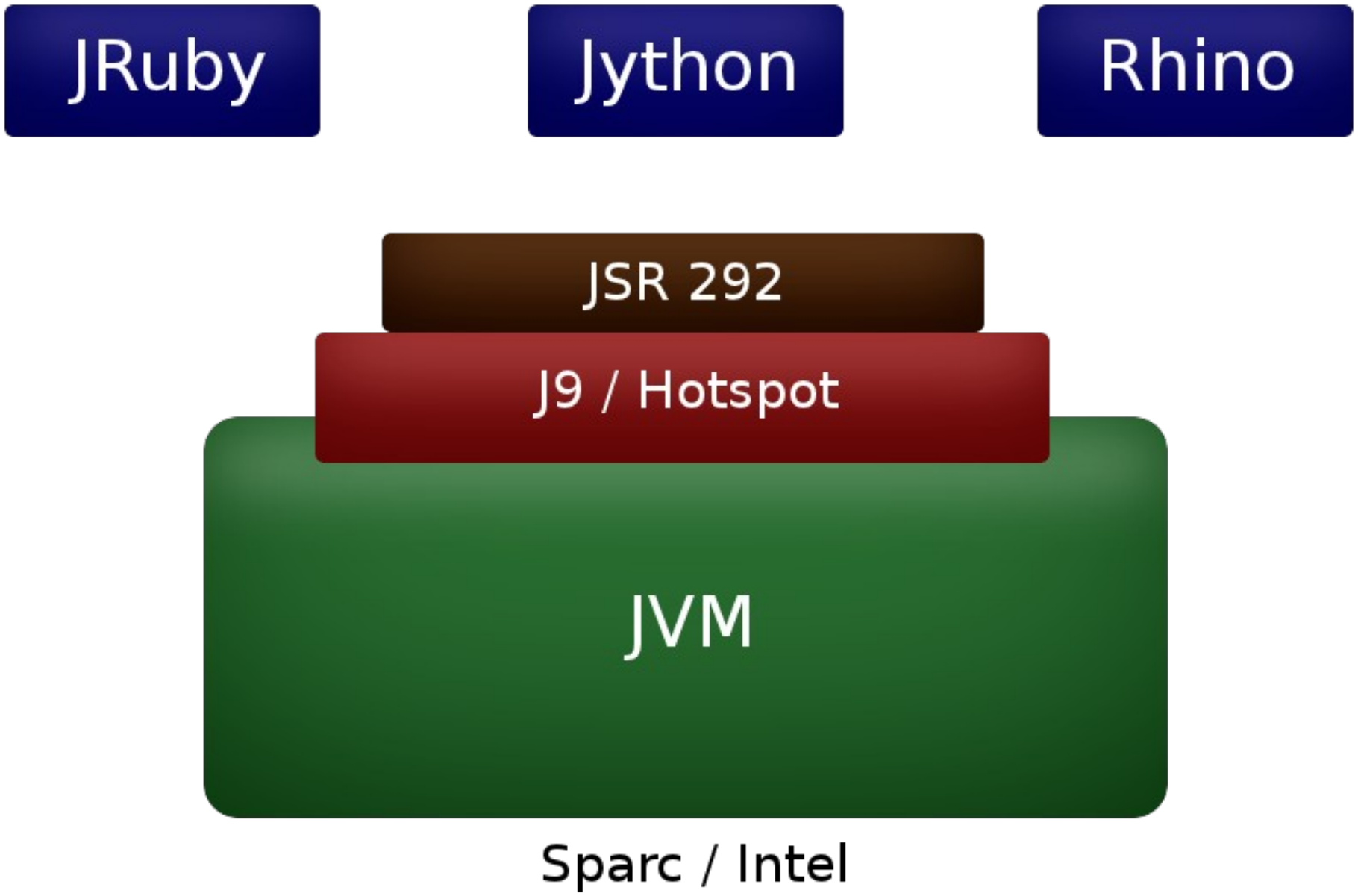
Sparc / Intel

JSR 292

Principe



JSR 292
Principe



OPEN WORLD FORUM

JSR 292
java.lang.invoke

- invokedynamic
  - édition de lien
- MethodHandle
  - invokeExact
  - invoke
  - combinateurs
  - GuardWithTest
  - ...
- MethodHandle Tree
  - combinaison de MH
- CallSite
  - conteneur de MH
- SwitchPoint
  - désoptimisation

JSR 292
Exemple

```
class A {
    foo() {
        ...
    }
}
```

.
.
.
.

```
...
b.foo()
...
```

invokevirtual java/lang/Object.foo:()Object

JSR 292
Exemple

```
class A {
    foo() {
        ...
    }
}
```

.
.
.
.

```
...
b.foo()
...
```

~~invokevirtual java/lang/Object.foo:()Object~~

invokedynamic "foo":()Object

JSR 292
Exemple

```
// exemple
…
A = B + C ;
...
```

```
…
invokedynamic "+" (Object, Object)Object
  bsm: RT.bsm(Lookup, String, MethodType)CallSite
...
```

JSR 292
Exemple

```
// exemple
…
A = B + C ;
...
```

```
…
invokedynamic "+" (Object, Object)Object
  bsm: RT.bsm(Lookup, String, MethodType)CallSite
...
```

1. appel methode d'amorce

```
CallSite bsm(Lookup lookup, String name, MethodType type) {
  CallSite cs = ...
  MethodHandle mh = ...
  cs.setTarget(mh);
  return cs;
}
```

JSR 292
Exemple

```
// exemple
…
A = B + C ;
...
```

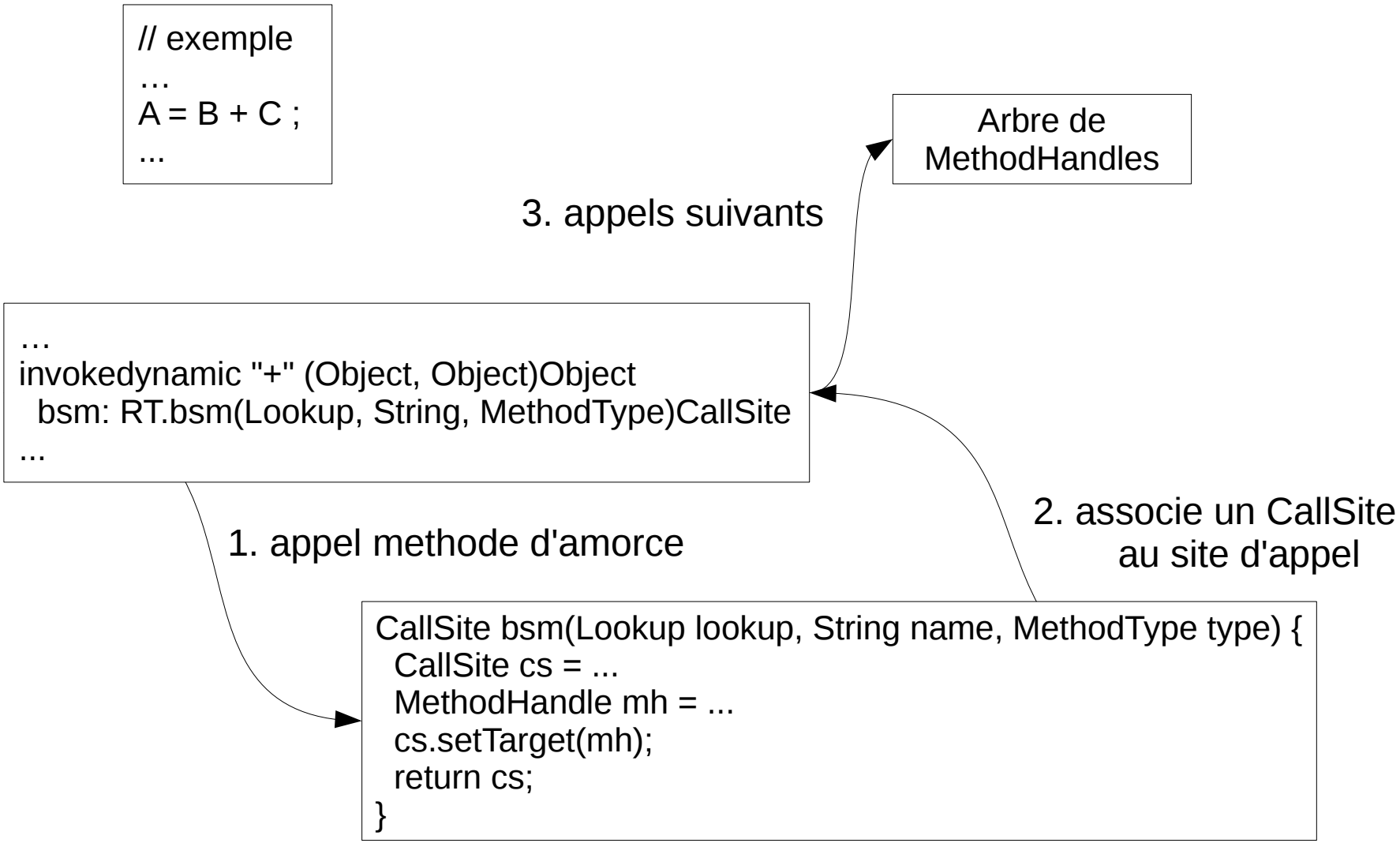
```
…
invokedynamic "+" (Object, Object)Object
  bsm: RT.bsm(Lookup, String, MethodType)CallSite
...
```

1. appel methode d'amorce

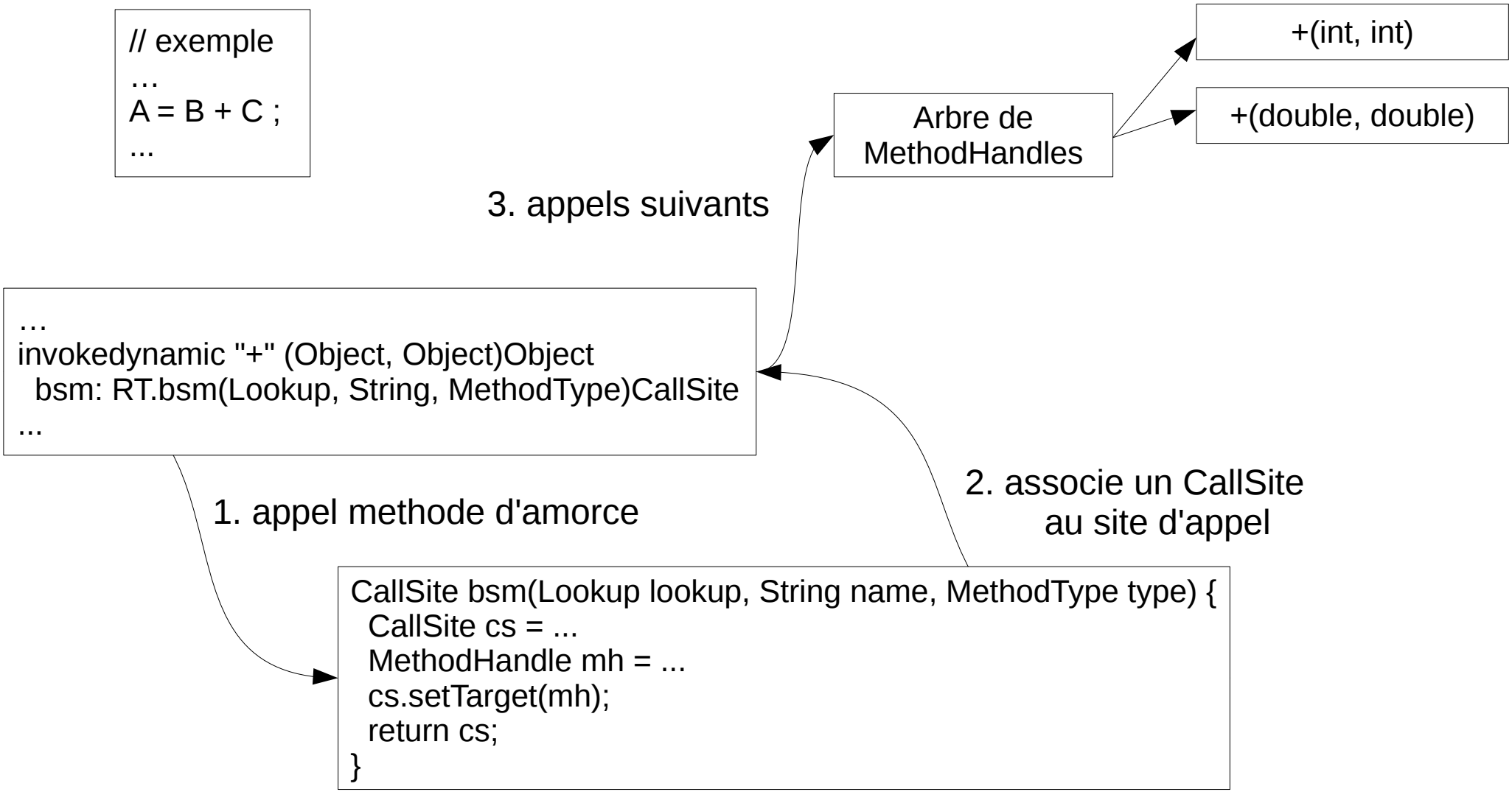
2. associe un CallSite au site d'appel

```
CallSite bsm(Lookup lookup, String name, MethodType type) {
  CallSite cs = ...
  MethodHandle mh = ...
  cs.setTarget(mh);
  return cs;
}
```

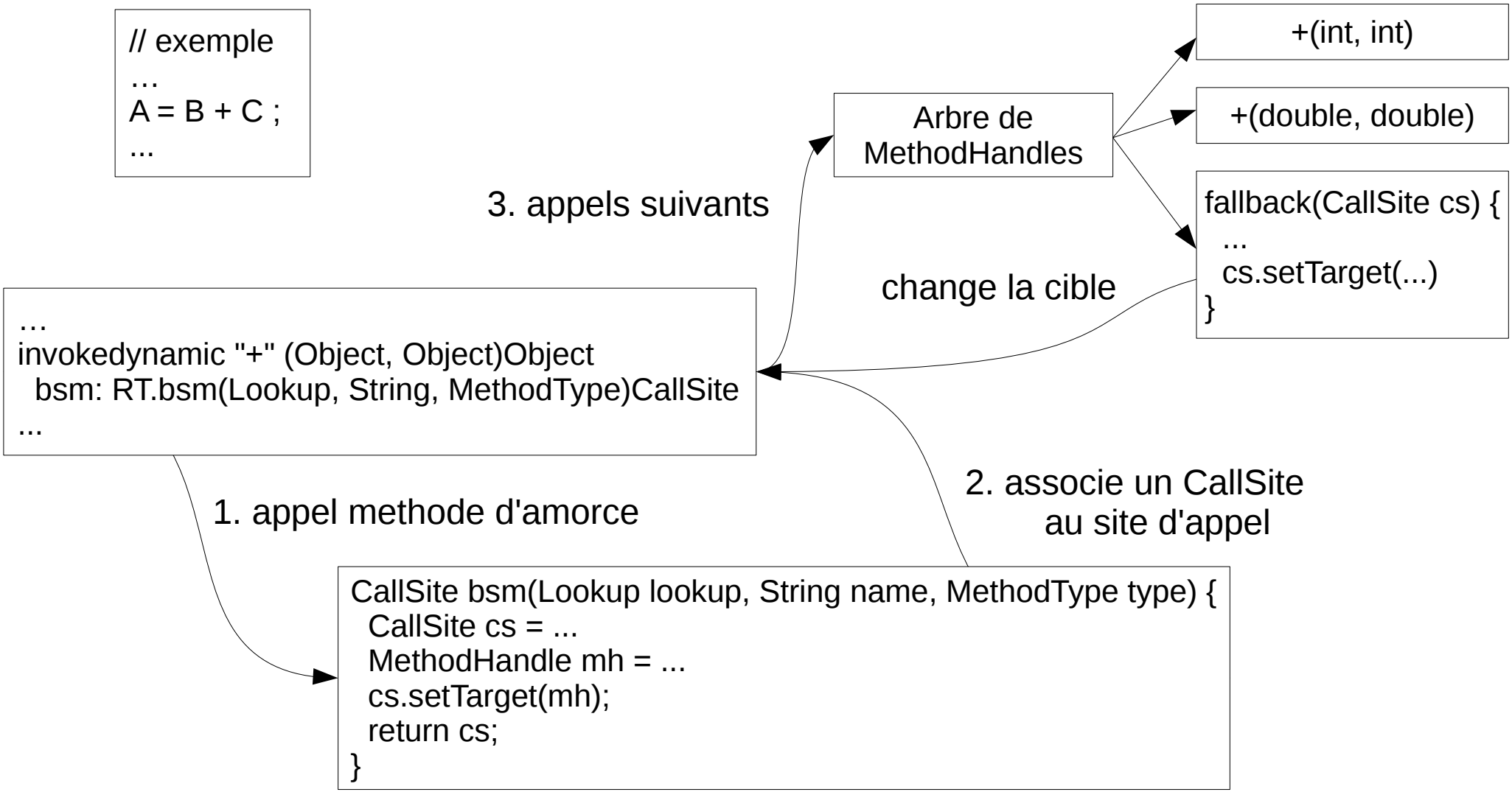
JSR 292
Exemple



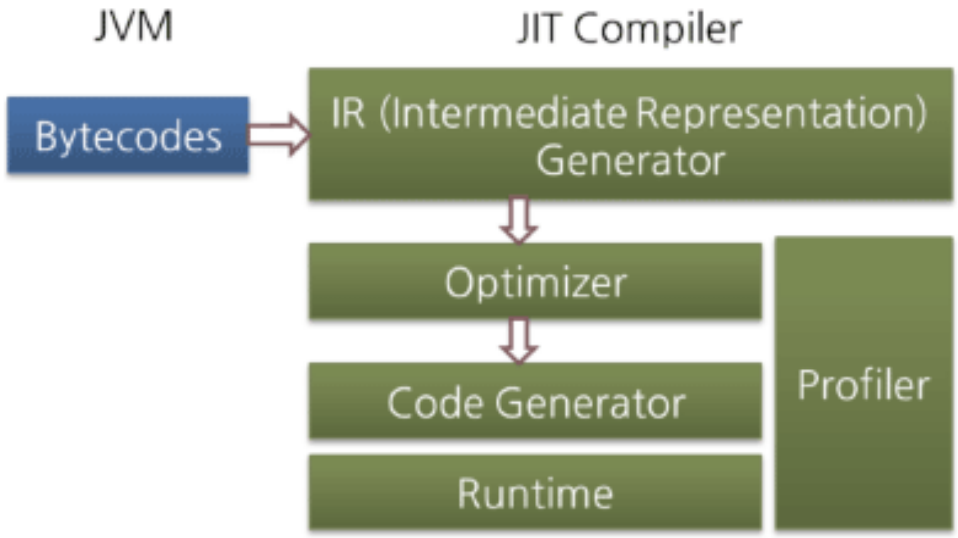
JSR 292
Exemple



JSR 292
Exemple



Optimisation
VMs modernes



- Profiling
  - présence de "null"
  - receveurs d'appels virtuels
  - classes pour les cast/instanceof
  - fréquence des branches des if
- Détection de code chaud
  - méthode chaude
  - boucle chaude (OSR)

- Génération de code machine optimiste
  - pas de test à null si jamais null
  - pas de branche non prise
  - inlining des appels virtuels
  - cheap-cast
  - désoptimisation si assomption fausse

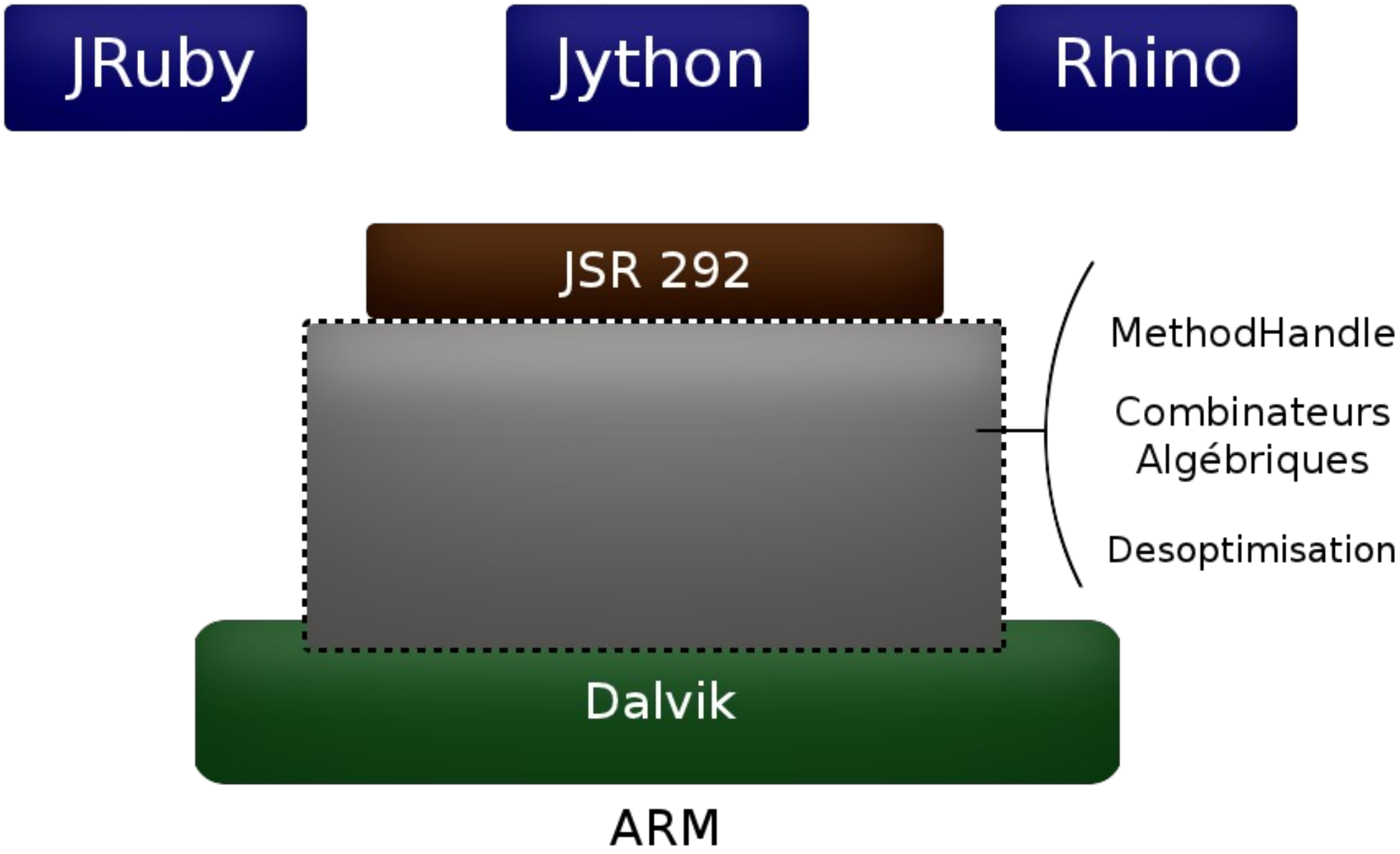
Optimisation JSR 292

- Profiling
  - détection de MethodHandle chaud et stabilité de invokedynamic
  - receveur de l'appel virtuel
  - branche pour le GuardWithTest
- Détection
  - stabilité target d'un invokedynamic
    - invokeExact / génération de bytecode
  - MethodHandle chaud
    - appel en assembleur
  - MethodHandles Tree
    - génération de bytecode => Out Of Memory !!!
    - (Pré-JDK 7) appel en Java
    - (JDK 7) mini-interpréteur (AST)
    - (JDK 8) LambdaForm + paramètres JITé

OPEN WORLD FORUM

Dalvik 292

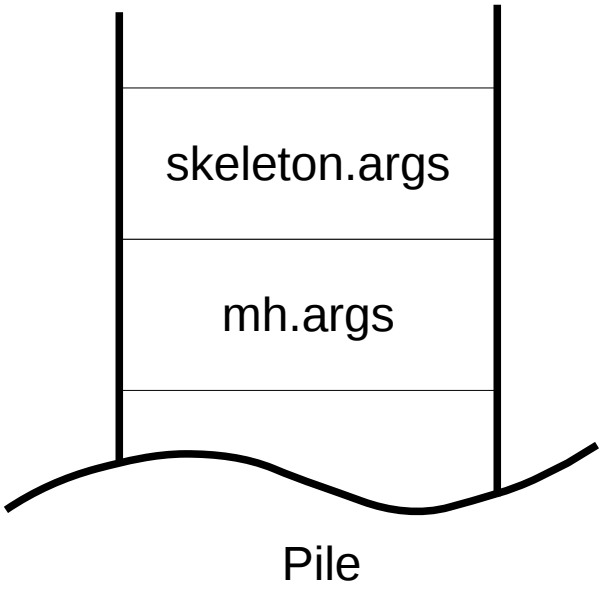
Problèmes et solutions



OPEN WORLD FORUM

Dalvik 292
Problèmes et solutions

- Détection de MethodHandle chaud
  - MethodHandle
    - bytecode spécifique
  - MethodHandle Tree
    - squelette en code DEX
    - convention d'appel spécifique
      - gère l'optimisation des lambdas (Java 8)
- JIT
  - MethodHandle et petit MethodHandle Tree
    - inlining
  - Squelette
    - JITé



Conclusion

- Avancement
  - ajout des instructions
  - nouveau format DEX
  - génération du fichier DEX
- Intérêt
  - Dalvik : une VM comme les autres
  - suivre évolution Java

OPEN WORLD FORUM

Dalvik 292

Jérôme Pilliet
<jerome.pilliet@univ-paris-est.fr>

Université Paris-Est de Marne-la-Vallée