Семантические инварианты проверяемые верификатором

- *Корректность стековых операций (у Java нетипизированный стек)*
- *Совместимость типов при записи в поля класса*
- *Доступ к локалам (локалы тоже нетипизированные)*
- *Использование методов в соответствии с сигнатурой (количество, тип)*
- *Доступ к методам и полям других классов*
- *Исполнение try/catch/finally блоков*

Лирическое отступление

```
static boolean finallyFun(boolean confuseJavac) {
        while (confuseJavac)
        {
              try {
                  return true;
              }
              finally {
                  break;
              }
          }
          return false;
      } *
```

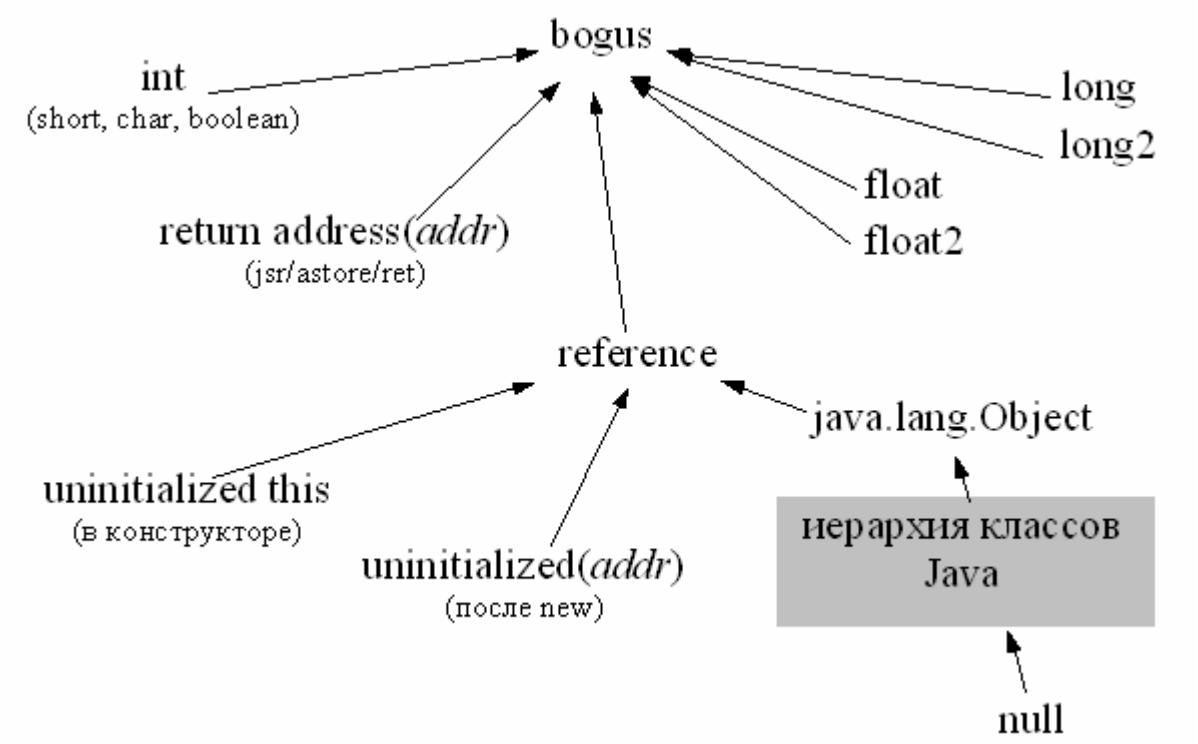
---

* - *пример Билла Веннерса*
http://www.artima.com/underthehood/finally2.html

Вопросы

- Можно ли в верифицируемый классфайл включить картинку? А исполняемый код x86?
- Осмысленна ли комбинация ACC_NATIVE | ACC_ABSTRACT, почему?
- Что бы вы добавили в формат классфайла?
- А что бы удалили (если обратная совместимость не проблема)?

Совместимость типов *



---

* *Картинка Кирилла Широкова*

Проверка корректности метода

Начальное состояние стека – пустое, локалы - сигнатура метода

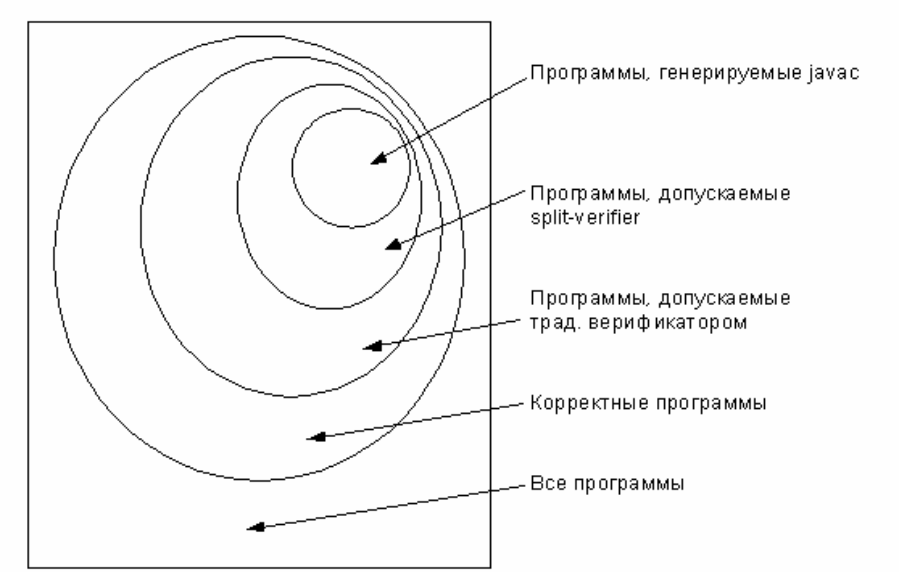
```
todo ← true
while todo = true do
  todo ← false
  for all i in all instructions of a method do
   if i was changed then
     todo ← true
   check whether stack and local variable types match definition of i
   calculate new state after i
   for all s in all successor instructions of i do
    if current state for s ≠ new state derived from i then
      assume state after i as new entry state for s
      mark s as changed
    end if
   end for
  end if
 end for
end while
```

Верификация

- Абстрактный интерпретатор оперирующий с типами (а не значениями), выполняет DFA
- Типичное время исполнения $O(n \log_2 t)$

  n – количество байткодов в методе,

  t – количество элементов в иерархии классов
- Возможно построение байткода с временем верификации $O(n^2)$



Двухфазная верификация

- Сотрудничество компилятора и JVM (описан в JSR 202, спецификация на Прологе)
- Новый атрибут *StackMapTable,* обязателен для классов версии начиная с 50 при наличии ветвлений или исключений
- Фиксированные точки DFA в начале базовых блоков
- Дальше абстрактная интерпретация
- Верификатор только проверяет корректность предоставленной информации
- Сложность времени исполнения $O(n)$

Пример неверифицируемого кода

```
/* только для старого  javac, порождающего jsr */
int test(boolean b) {
   int i;        /* не инициализировано */
   try {
      if (b) return 1;
      i = 2;
    } finally {  /* при входе сюда i может быть не определено */
      if (b) i = 3;  /* но мы его определяем */
     }
    return i;  /* и, возможно, возвращаем */
}
```

Вопросы

- Почему верификация не заменена подписыванием кода?
- Что верификатор может сообщить JITу?
- Попробуйте построить примеры элементы дополнений со слайда 3
- Сколько фактических ошибок вы найдёте в этом тексте http://www.insidepro.com/kk/215/215r.shtml ?