情報システムプロジェクト1

第3回

Var.java, VarTable.java

2023年4月26日

今日の実習内容

- VSMの動作の理解
- VSMアセンブラへの手動コンパイル
- Var.java, VarTable.java の作成

Var クラス, VarTable クラス

Var		# 変数定義部
- type	: Type	# 変数の種類
- name	: String	# 変数の名前
- address	: int	# 変数のDseg上のアドレス
- size	: int	# サイズ
Var (type : Type, name : String, addr : int)		# コンストラクタ
getType ()	: Type	# 変数の種類を返す
getName ()	: String	# 変数の名前を返す
getAddress ()	: int	# 変数のアドレスを返す
getSize ()	: int	# 変数のサイズを返す

VarTable		
- varList	: ArrayList<Var>	# 変数表
- nextAddress	: int	# 次の変数のアドレス
VarTable ()		# コンストラクタ
- getVar (name : String)	: Var	# 変数を返す
exist (name : String)	: boolean	# 変数の存在判定
registerNewVariable (type : Type, name : String, size, int)	: boolean	# 変数表に要素追加
getAddress (name : String)	: int	# アドレスを返す
getType (name : String)	: Type	# 種類を返す
checkType (name : String, type : Type)	: boolean	# 型の一致判定
getSize (name : String)	: int	# 変数のサイズを返す
size ()	: int	# 表のサイズを返す
removeTail (index : int)	: void	# 表の末尾を削除する

変数表への挿入

- 変数表への挿入

```
/** @ return 変数 name を登録できたか？ */
boolean registerNewVariable
    (Type type, String name, int size)
```

例：int i, j;

型は INT

スカラ変数の
サイズは 1

```
varTable. registerNewVariable (Type.INT, “i”, 1);
varTable. registerNewVariable (Type.INT, “j”, 1);
```

例：int a[5], b[] = {1, 2, 3};

型は ARRAYOFINT

配列の
サイズ

```
registerNewVariable (Type.ARRAYOFINT, “a”, 5);
registerNewVariable (Type.ARRAYOFINT, “b”, 3);
```

VarTable.java

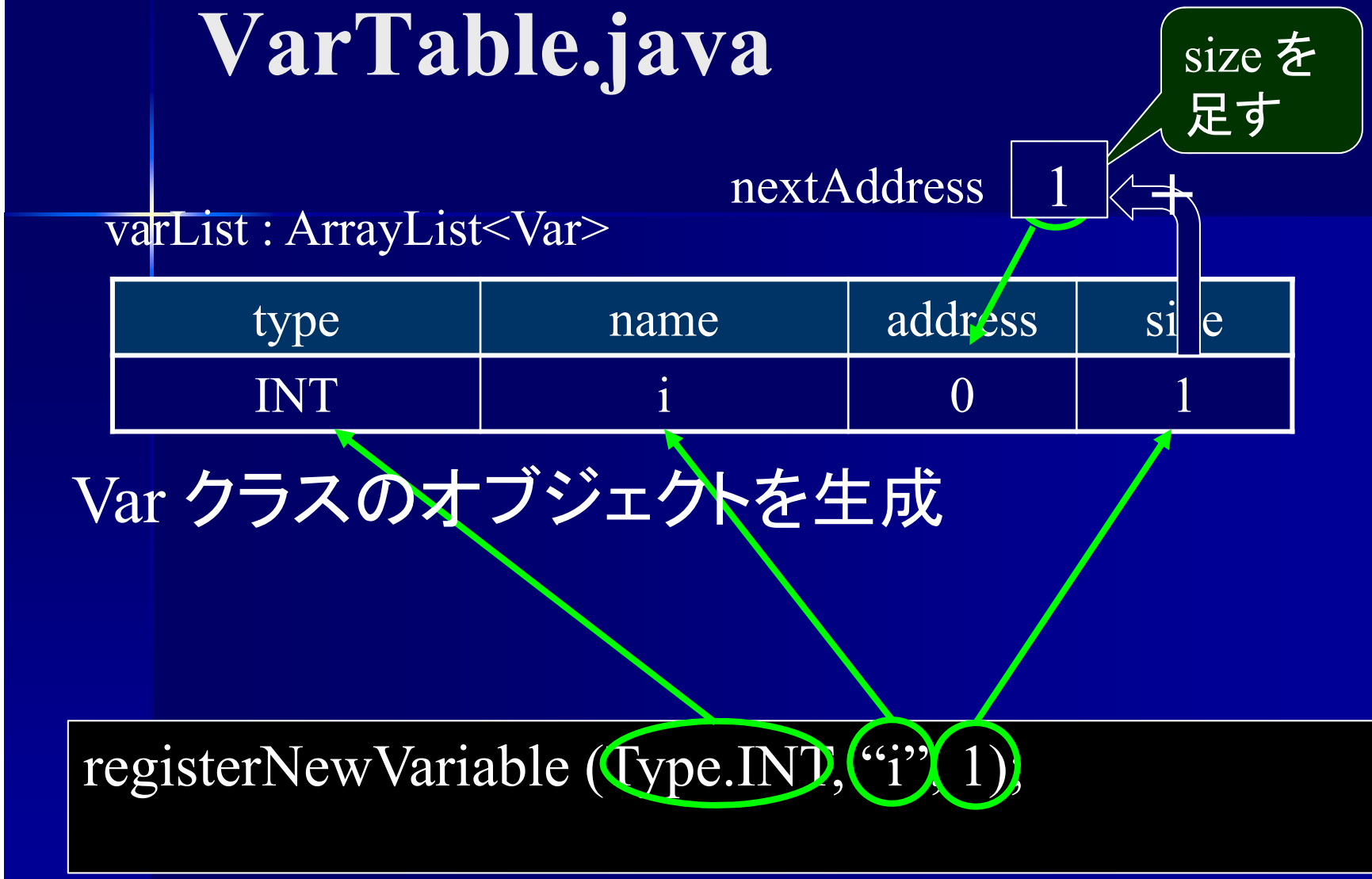
nextAddress | 0 | 初期値 0

varList : ArrayList<Var>

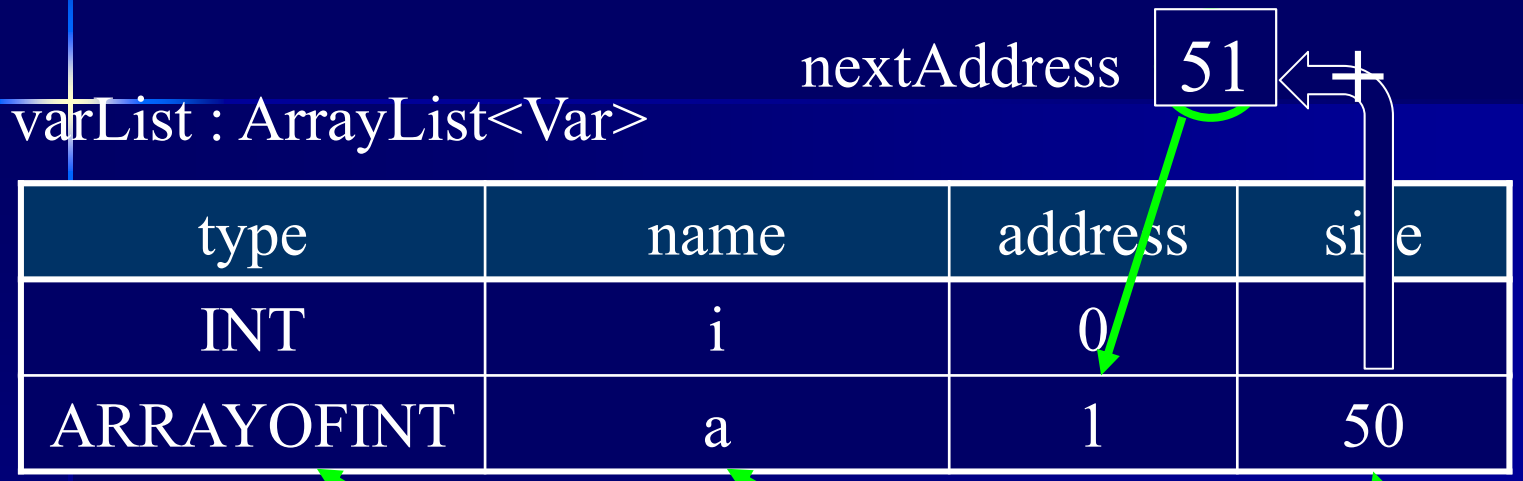
type	name	address	size

空の ArrayList

VarTable.java



VarTable.java




```
registerNewVariable (Type.INT, “i”, 1);
registerNewVariable (Type.ARRAYOFINT, “a”, 50);
```

変数表への登録判定

- 変数表への登録判定は
  VarTable.exist (String) を使用

```
/** @ return 変数 name は登録済か？ */
boolean exist (String name)
```

例 : 変数 x は登録済か？

```
varTable.exist (“x”)
```

変数の型判定

- 変数の型判定は

  VarTable.checkType (String, Type) を使用

```
/** @ return 変数 name の型が type か？ */
boolean checkType (String name, Type type)
```

例 : 変数 i は int 型か？

```
varTable.checkType (“i”, Type.INT)
```

変数の番地

- 登録された変数の番地を得るには
  VarTable.getAddress (String) を使用

```
/** @ return 変数 name の番地 */
int getAddress (String name)
```

登録されていない変数の場合は返り値は -1

例 : 変数 i の番地

```
varTable.getAddress (“i”)
```

VarTable.java

nextAddress 152

varList : ArrayList<Var>

type	name	address	size
INT	i	0	1
ARRAYOFINT	a	1	50
ARRAYOFINT	b	51	100
INT	n	151	1

exist (“n”) ⇒ true

checkType (“i”,Type.INT) ⇒ true

checkType (“x”,Type.ARRAYOFINT) ⇒ false

VarTable.java

nextAddress 152

varList : ArrayList<Var>

type	name	address	size
INT	i	0	1
ARRAYOFINT	a	1	50
ARRAYOFINT	b	51	100
INT	n	151	1

getAddress (“i”) ⇒ 0

getAddress (“b”) ⇒ 51

getAddress (“x”) ⇒ -1

変数表のサイズ

- 変数表のサイズ(登録されている変数の個数)
  VarTable.size () を使用

```
/** @ return 変数表のサイズ */
int size ()
```

VarTable.java

nextAddress 202

varList : ArrayList<Var>

type	name	address	size
INT	i	0	1
ARRAYOFINT	a	1	50
ARRAYOFINT	b	51	100
INT	n	151	1
ARRAYOFINT	c	152	50

size() $\Rightarrow$ 5

変数表からの削除

- 変数表の末尾に登録された変数を削除するには
  VarTable.removeTail (int) を使用

```
/** index 番目以降の変数を削除 */
void removeTail (int index)
```

登録されている変数の個数以上の
値を指定した場合は何もしない

例 : 5番目以降の変数を削除

```
varTable.removeTail (5)
```

VarTable.java

nextAddress 151

varList : ArrayList<Var>

	type	name	address	size
0	INT	i	0	1
1	ARRAYOFINT	a	1	50
2	ARRAYOFINT	b	51	100
3	INT	n	151	1
4	ARRAYOFINT	c	152	50

removeTail (3)

VarTable.java

nextAddress 151

varList : ArrayList<Var>

	type	name	address	size
0	INT	i	0	1
1	ARRAYOFINT	a	1	50
2	ARRAYOFINT	b	51	100

removeTail (3)

VarTable.java

nextAddress 151

varList : ArrayList<Var>

	type	name	address	size
0	INT	i	0	1
1	ARRAYOFINT	a	1	50
2	ARRAYOFINT	b	51	100

removeTail (10) ⇒ 何もしない

removeTail (-1) ⇒ 何もしない