

Java基礎講座

- きしだ なおき

2007/6/27

自己紹介

- 福岡でフリーでやっています
- 創るJavaっていう入門書、書いてます。

今日の予定

- 言語の話
- 実行の話
- GUIの話

方針

- 入門ではなくて基礎
- Javaの仕組みを知って理解につなげる

言語の話

- 基本型と参照型
- クラスの話

基本型と参照型

- 基本型
 - 変数が値を持つ
- 参照型
 - 変数はオブジェクトへの参照を持つ
 - オブジェクトは「ヒープ」に入れられる

クラスの話

- Javaはクラス指向
- staticと非static
- メソッド内部クラスとfinal変数

Javaはクラス指向

- オブジェクトよりも、変数の型が大切
- オブジェクトのメンバは属するクラスに従う

staticと非static

- static
 - クラス名.xxxで使える
- 非static
 - (newしたオブジェクト).xxxで使える
- フィールドの場合
- メソッドの場合
- クラスの場合

static:フィールドの場合

- staticフィールド
 - クラスが持つ
- 非staticフィールド
 - newしたときに領域確保される

```
class A{  
    int a;  
    static b;  
}
```

static: メソッドの場合

- static
 - オブジェクト関係なし
 - オーバーライドできない
- 非static
 - 暗黙の引数thisを持つ
 - オーバーライドできる

```
class A{  
    int a;  
    void m(){  
        System.out.println("やあ" + this.a);  
    }  
}
```

```
class A{  
    int a;  
    static void m(A _this){  
        System.out.println("やあ" + _this.a);  
    }  
}
```

static: クラスの場合

- 入れ子になったクラス
 - Nested Class
 - 名前に外側クラスの名前が付けられる

非static

インナークラス
外側のクラスの
オブジェクトを保持する

static

名前が長いだけ

インナークラスとstaticクラスの関係

•インナークラス

```
class Outer{
  class Inner{
    void hoge(){
      bar();//Outerクラスのメソッド
    }
  }
}

void foo(){
  new Inner();
}

void bar(){
}
}
```

•staticクラス

```
class Outer{
  static class Inner{
    Outer outer;//外側クラスへの参照
    Inner(Outer outer){//コンストラクタで受け取る
      this.outer = outer;
    }
  }
  void hoge(){
    outer.bar();//Outerクラスのメソッド
  }
}

void hoge(){
  new Inner(this);//コンストラクタに自分を渡す
}

void bar(){
}
}
```

メソッド内部クラス

- 利用するローカル変数をフィールドにもつ

```
void a(){
    final String str = input.getText();
    class Runner implements Runnable{
        public void run(){
            textarea.append(str + "\n");
        }
    }
    SwingUtilities.invokeLater(new Runner());
}
```

```
void a(){
    final String str = input.getText();
    SwingUtilities.invokeLater(new Runner(str));
}

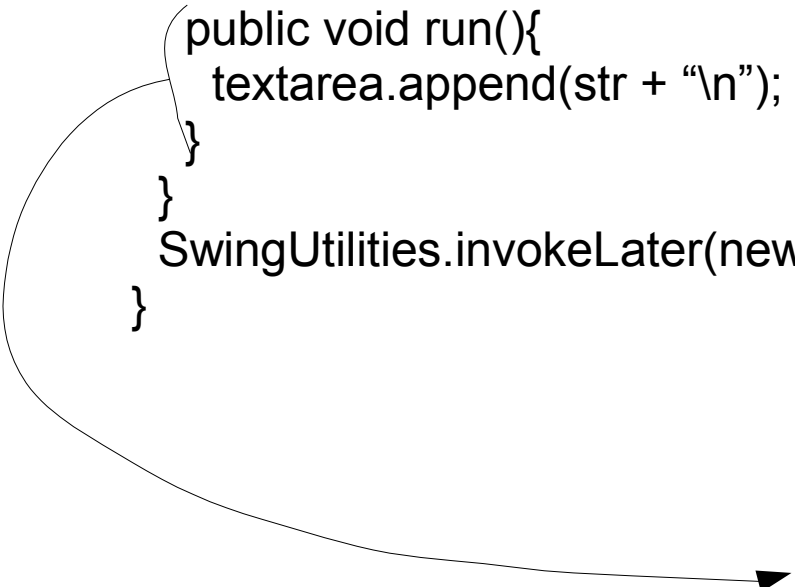
class Runner implements Runnable{
    String str;
    Runner(String str){
        this.str = str;
    }
    public void run(){
        textarea.append(str + "\n");
    }
}
```

匿名クラス

- newするときについてメソッドオーバーライド

```
void a(){
    final String str = input.getText();
    class Runner implements Runnable{
        public void run(){
            textarea.append(str + "\n");
        }
    }
    SwingUtilities.invokeLater(new Runner());
}
```

```
void a(){
    final String str = input.getText();
    SwingUtilities.invokeLater(new Runnable(){
        public void run(){
            textarea.append(str + "\n");
        }
    });
}
```



Javaの実行

- メモリの話
- プログラムが動くまで

メモリの話

- Javaのメモリ
- スタックとヒープ
- ガーベージコレクション

Javaのメモリ

- ネイティブ領域
 - VMが使う
- Javaヒープ領域
 - オブジェクトが入る
- PermGen領域
 - クラス、メソッドのバイトコードや定数

スタックとヒープ

- スタック
 - ローカル変数
 - メソッドの戻り先
 - スコープを抜けるときに破棄
- ヒープ
 - オブジェクトが保持するデータ
 - フィールド

ガベージコレクション

- 不要なオブジェクトを回収
- 世代別GC

参考

http://www.atmarkit.co.jp/fjava/rensai3/javavm02/javavm02_1.html

不要なオブジェクトを回収

- ローカル変数から参照をたどる
- 全部の参照をたどったとき、どこからも参照されてなければゴミ

世代別GC

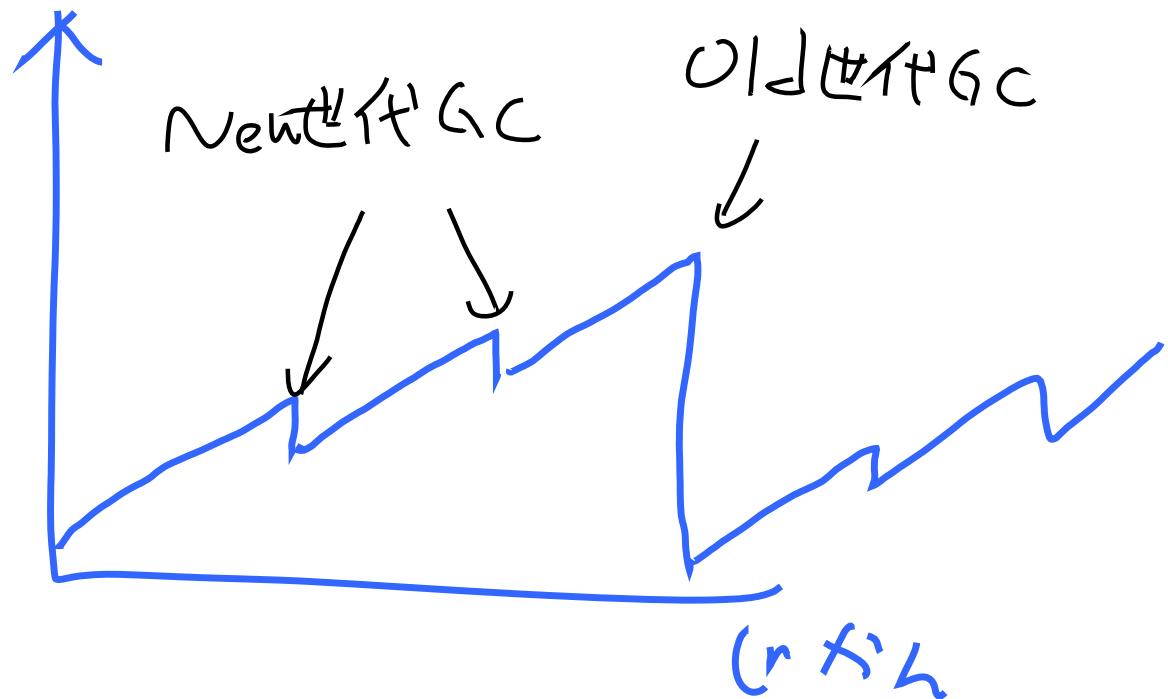
- 領域分け

- New領域

- Eden領域
 - From領域
 - To領域

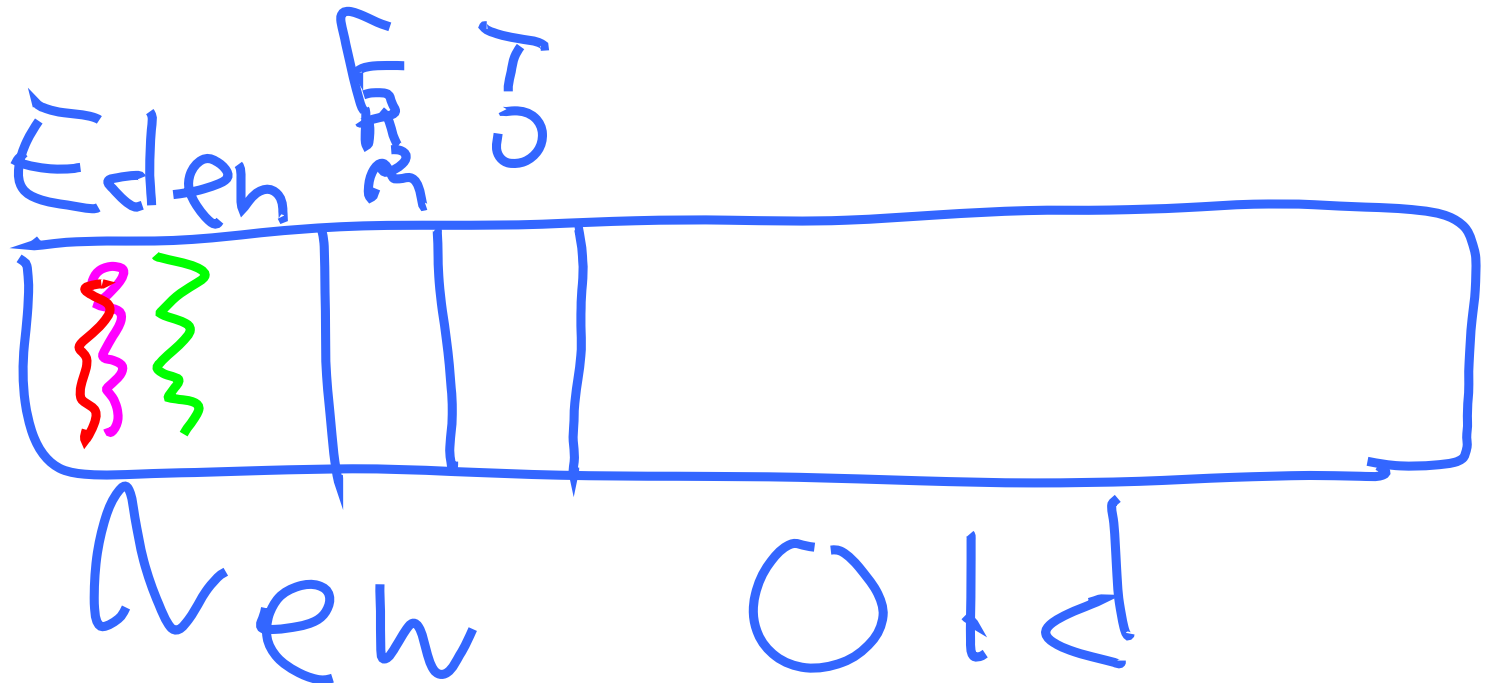
- Old領域

メモリ



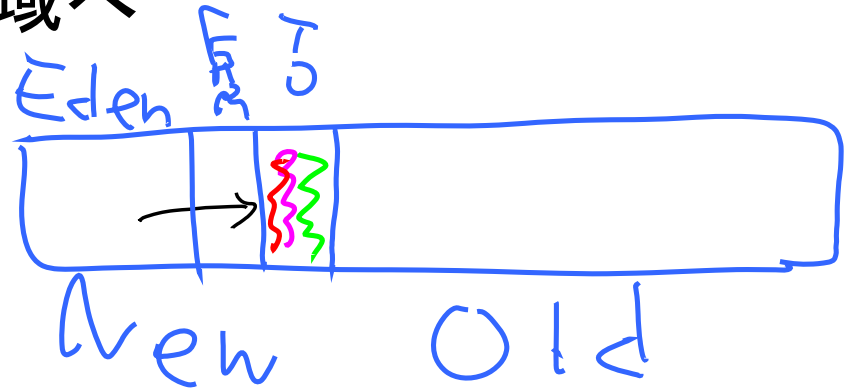
世代別GCの手順

- オブジェクトの生成
 - まずEden領域へ
 - いっぱいになったらNew世代GC

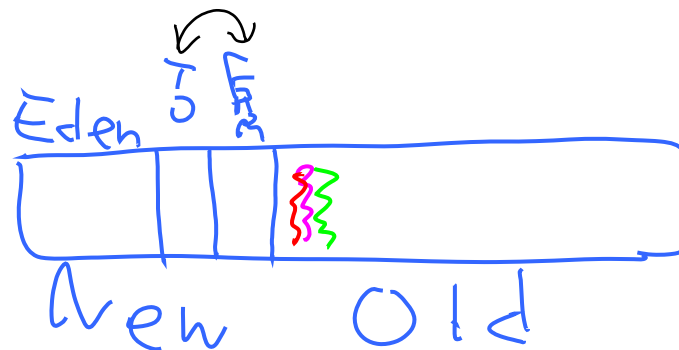


New世代GC

- 生きてるオブジェクトをTo領域へ
- プログラムの実行は停止
- FromとToを入れ替え
- 2回目以降

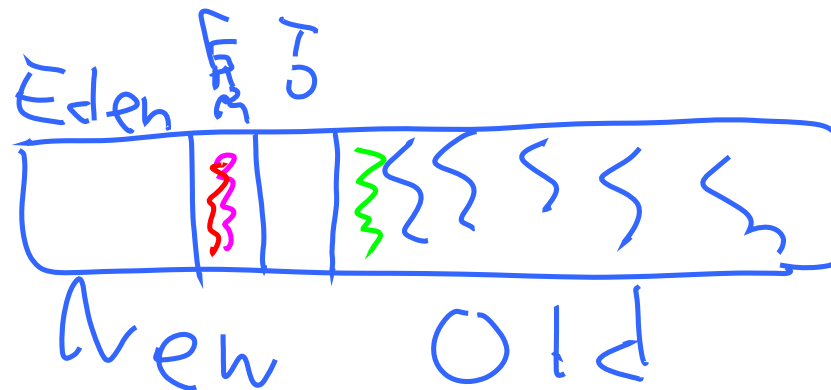


- (さっきまでTo領域だった)From領域の生きてるオブジェクトをTo領域へ
- このときオブジェクトの歳が増える
- ある程度歳をとったオブジェクトをOld領域へ



Old世代GC

- mark-sweep-compact
 - mark
 - 生きているオブジェクトに印をつける
 - sweep
 - 不要なオブジェクトを回収
 - compact
 - 生きているオブジェクトを一箇所に集める



プログラムが動くまで

- コンパイル
 - コンパイル時の最適化
- バイトコードを解釈しながら実行
 - 実行時の最適化

コンパイル時の最適化

- Javaコードがそのままの形でバイトコードになるとは限らない
- for(;;)がどうなるか

※do~whileになるという話をしましたが、実際には条件なしの無限ループになるようです
逆コンパイラで確認していたのですが、条件なしの無限ループは対応するJavaコードがないために逆コンパイラがdo~whileで表示していたということです。

実行時の最適化

- バイトコードの解釈
 - JIT
 - 実行時にあらかじめネイティブコンパイル
 - HotSpot
 - よく実行されるところだけネイティブコンパイル
- オブジェクトの配置
 - ヒープの欠点
 - GCが必要
 - 遠くにあるのでキャッシュが効かない
 - メモリはまとまってるほうが効率がいい
 - ローカルでしか使わないオブジェクトはスタックへ

http://www.atmarkit.co.jp/fjava/rensai3/javavm01/javavm01_2.html

http://www-06.ibm.com/jp/developerworks/java/051104/j_j-jtp09275.shtml

GUIの話

- AWTとSwingの動きの違い
- 歴史的なこと

動きの違い

- AWT/SWTは、ネイティブのコンポーネントを利用
- SwingはJavaでがんばる

GUIの歴史

- 間に合わせでAWT
- がんばってSwing
- 不満爆発SWT
- くやしくってSwing