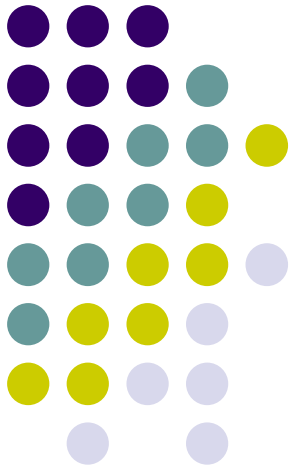
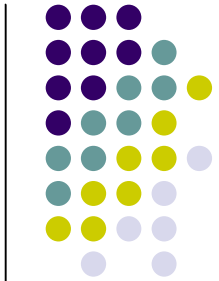
Javaゼミ第二回目

2.8 StringBuffer クラス
2.9 オブジェクトの配列
2.10 コマンドライン引数
2.11 System クラス



発表者：佐々木研　加藤友宏

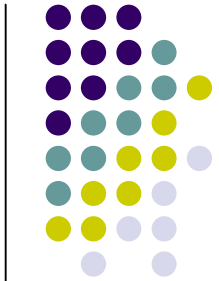
2.8 StringBufferクラス



- StringBufferクラスとは？
  - ・文字列をカプセル化
  - ・文字を追加、挿入するメソッドがある

→Stringクラスで出来なかったカプセル化された
文字列を変更することが可能

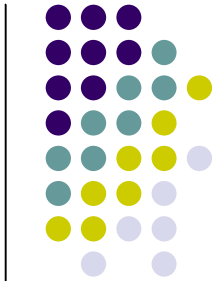
2.8 StringBufferコンストラクタ



- StringBuffer()
  ・バッファ容量を16文字で初期化
- StringBuffer(int size)
  ・指定した文字数で初期化
- StringBuffer(String s)
  ・sの内容で初期化して、拡張用に16文字確保

※内部バッファはオーバーフローしたら、
自動的にバッファ容量が増える。

2.8 StringBufferクラスの例



- StringBufferオブジェクトを3つのコンストラクタで生成し、
  各バッファの現在の容量とサイズを表示するプログラム。

```
class StringBufferDemo {
  public static void main(String args[]) {
    StringBuffer sb1 = new StringBuffer();                  // 16文字分のバッファ
    StringBuffer sb2 = new StringBuffer(30);                // 30文字分のバッファ
    StringBuffer sb3 = new StringBuffer("abcde");           // abcdeの5文字+16文字分のバッファ
    System.out.println(“sb1.capacity = ” +sb1.capacity());  //sb1の容量を表示
    System.out.println(“sb1.length = ” + sb1.length());     //sb1のサイズを表示
    System.out.println(“sb2.capacity = ” + sb2.capacity()); //sb2の容量を表示
    System.out.println(“sb2.length = ” + sb2.length());     //sb2のサイズを表示
    System.out.println(“sb3.capacity = ” + sb3.capacity()); //sb3の容量を表示
    System.out.println(“sb3.length = ” + sb3.length());     //sb3のサイズを表示
  }
}
```

実行結果

```
>java StringBufferDemo
sb1.capacity = 16
sb1.length = 0
sb2.capacity = 30
sb2.length = 0
sb3.capacity = 21
sb3.length = 5
```

2.8 StringBufferの文字の追加

- appendメソッドとinsertメソッド
  - ・どんな種類のデータも受け取ることが可能
  - ・append メソッドは常に、バッファの末尾に与えられた文字を追加

    StringBuffer append(※1)
  - ・insert メソッドは指定された位置に文字を追加

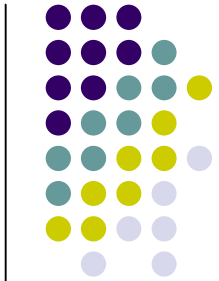
    StringBuffer insert(int i,※2)

※1：boolean,char,double,float,int,long,Object,String。

※2：boolean,char,int,long,Object,String

Insertは、
doubleとfloatが使えない

2.8 文字列追加の例

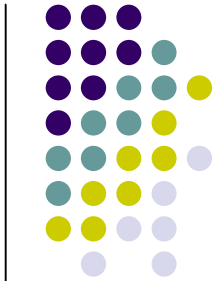


仮定：zを、"start"を含む文字列バッファオブジェクトとみなす。

・z.append("le")とすると、文字列バッファの内容が"startle"と変更される。

・z.insert(4,"le")とすると、文字列バッファの内容が"starlet"と変更される。

2.9オブジェクトの配列

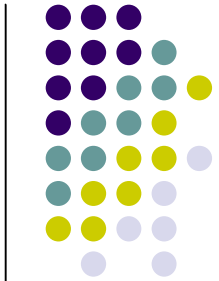


- 基本データ型の配列のオブジェクト版
- 1次元でも多次元でもOK
- 作成手順
  1. 配列を宣言する
  2. 配列要素のスペースを確保する
  3. 要素の初期設定

例：String array[] = new String[1];

```
array[0] = "String 0";
```

2.9 オブジェクトの配列の例



- 配列を作成し、表示するプログラム

```
class StringArray {
  public static void main(String args[]) {
    String array[] = new String[3];       //配列の宣言
    array[0] = "String 0";                //配列array[0]を"String 0"で初期化
    array[2] = "String 2";                //配列array[2]を"String 2"で初期化
    System.out.println(array.length);  //配列の個数を表示
    System.out.println(array[0]);         //配列array[0]の内容を表示
    System.out.println(array[1]);         //配列array[1]の内容を表示
    System.out.println(array[2]);         //配列array[2]の内容を表示
  }
}
```

実行結果

```
>java StringArray
3
String 0
null
String 2
```

明示的に初期化されていないので、
オブジェクトを参照していないため、nullになっている

2.10　コマンドライン引数

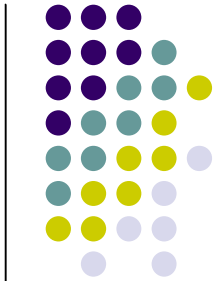
- main()・・・静的メソッドでStringオブジェクトの配列の引数を一つ取る。

main(String args[])

入力した引数を表す。

- 個数は、int型のargs.length。
- 引数はargs[0],args[1],args[2],など。

2.10 コマンドライン引数の例



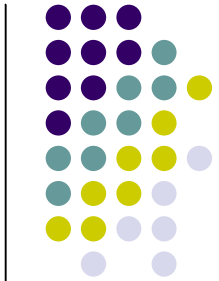
- コマンドライン引数の個数と内容を表示するプログラム。

```
class CommandLine {                          ← Stringオブジェクト
  public static void main(String args[]) {
    System.out.println("args.length = " + args.length);   ← 個数を表示
    System.out.println("args[0] = " + args[0]);           ← 一番目の引数を表示
    System.out.println("args[1] = " + args[1]);           ← 二番目の引数を表示
  }
}
```

実行結果

```
>java CommandLine 1 2
args.length = 2
args[0] = 1
args[1] = 2
```

2.11 Systemクラス

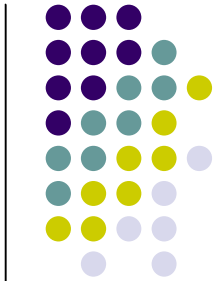


- ランタイム環境に関するいくつかの属性が定義されている。

静的変数	説明	参照オブジェクト
out	「標準」出力ストリーム print(),println()で表示	PrintStream
err	「標準」エラー出力ストリーム	
in	「標準」入力ストリーム	InputStream

表1：クラスフィールドの概要

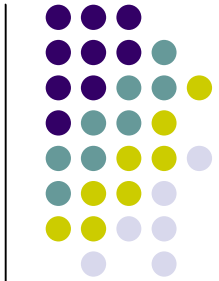
2.11 Systemクラス



静的メソッド	説明
void exit(int code)	プログラムの終了 codeが0なら正常終了 それ以外は、問題発生時
long currentTimeMillis()	グリニッジ標準時からの 経過時間（ミリ秒）
void arraycopy(Object a, int aIndex, Object b, int bIndex, int size)	2つの配列間での要素のコピー aが元の配列 bがコピー先の配列

表2：メソッドの概要

2.11 Systemクラス



- 配列array1から配列array2へ要素をコピーするプログラム

```
class ArrayCopy {
  public static void main(String args[]) {
    int array1[] = { 0, 1, 2 };                    //コピー元の配列
    int array2[] = { 0, 0, 0 };                    //コピー先の配列
    System.arraycopy(array1, 0, array2, 0, 2); // 配列をコピー
    System.out.print("array2: ");
    System.out.print(array2[0] + " ");
    System.out.print(array2[1] + " ");
    System.out.println(array2[2]);
  }
}
```

配列array1の最初から2つ分を
配列array2の最初にコピーする

実行結果

```
>java ArrayCopy
array2: 0 1 0
```

2つ分しかコピーしていないので、
array2[2]の要素はコピーされない。

練習問題

- 以下のコードと同等なコードをStringBufferを用いて作成し、結果を表示せよ。

  x = "a" + 4 + "c";

- コマンドラインから2つのdouble型引数を受け取り、その四則演算を表示するプログラムを作成せよ。
- グリニッジ標準時を用いて、現在の年を表示するプログラムを作成せよ。