Programming in JAVA

lecture 3

Logic operators, tables and advanced string processing

Logic operators - example

```
Scanner sc = new Scanner(System.in);

System.out.print("Enter x : ");
float x = sc.nextFloat();
System.out.print("Enter y : ");
float y = sc.nextFloat();

if ( (x>=0)&&(y>=0) ) {
   System.out.println("point is in I quadrant");
        } else if ( (x<0)&&(y>0) ) {
   System.out.println("point is in II quadrant");
        } else if ( (x<=0)&&(y<=0) ){
   System.out.println("point is in III quadrant");
        }else {
   System.out.println("point is in IV quadrant");
        }
```

Tables - declaration examples

```
// Method A:
   float[] tableA;
   table = new float [5];

// Method B:
   float[] tableB = new float [5];


// Method C:
   float[] tableC ={ 1.0f, 2.0f, 3.0f, 4.0f, 5.0f };
```

Tables – examples

```
for (int i = 0; i < tableC.length; ++i)
      System.out.println("tableC[" + i + "] = " +
                                tableC[i]);


for (float el : tableC)
       System.out.println("tableC[...] = " +
                                 el);
```

Bubble sort in JAVA

```
int[] table = {4,7,-1,1,6,7,1,8,-3,-5};


int tmp;
boolean flip=true;
while (flip)
{
     flip=false;
     for(int i=0; i<table.length-1 ; i++)
     if (table[i]>table[i+1]) {
              flip=true;
              tmp = table[i];
              table[i]=table[i+1];
              table[i+1]=tmp;
     }
}
```

2-dimension table

```
int n=10;

/* declaration of 2-dimensional tables (matrix) */
int[][] tab2D = new int[10][10];


/* filling the matrix with numbers */
    for(int i=0; i<n ; i++)
       for(int j=0; j<n ; j++)
          tab2D[i][j]=i+j+1;
```

2-dimension table, part 2

```
/* printing out a 2-dimensional table */
for(int i=0; i<n ; i++) {
   for(int j=0; j<n ; j++)
       System.out.print("\t"+tab2D[i][j]+",");
   System.out.println();
}
```

2-dimension table, part 3

```
/* setting zeroes on the diagonal */
    for(int i=0; i<n ; i++)
       tab2D[i][i]=0;

/* setting zeroes on the second diagonal */
    for(int i=0; i<n ; i++)
       tab2D[i][n-i-1]=0;
```

2-dimension table, column switch

```
Scanner sc = new Scanner(System.in);
System.out.print("Enter index of column 1 : ");

int index1 = sc.nextInt();
System.out.print("Enter index of column 2 : ");
int index2 = sc.nextInt();

int tmp;
for(int i=0; i<n ; i++){
     tmp=tab2D[i][index1];
     tab2D[i][index1]=tab2D[i][index2];
     tab2D[i][index2]=tmp;
}
```

function String.format

```
int i = 9;

double d0 =   45.33454d;
double d1 = 123.234578d;
double d2 =   3.232765d;
String str = "abc";

String result = String.format("i = %d ; d = %f ;
                           str = %s", i, d0, str);

String result2 = String.format("d0 =%8.2f %nd1
                   =%8.2f %nd2 =%8.2f", d0,d1,d2);


System.out.println(result);
System.out.println(result2);
```

Caesar cipher implementation as an example for string processing

```
String alphabet = "ABCDEFGHIJKLMNOPRSTUWVXYZ ";

String message = "MESSAGE TO CODE";

String cipher="";

int key =3;

int tmp;
```

Caesar cipher - coding

```
for(int i=0; i<message.length();i++)
{
      tmp = alphabet.indexOf(message.charAt(i));
      tmp += key;
      tmp = tmp%alfabet.length();
      cipher += alphabet.charAt(tmp);
}
```

Caesar cipher - decoding

```
String decrypted_message ="";
for(int i=0; i<cipher.length();i++)
{
   tmp = alphabet.indexOf(cipher.charAt(i));
   tmp -= key;
   tmp = (tmp+alphabet.length())%alphabet.length();
   decrypted_message += alphabet.charAt(tmp);
}


System.out.println("decrypted_message =”
                   +decrypted_message);
```