Repetitive structures or loops

- **while (***expression***)** *statement*
- Activity: make a program to count down till 0, the user gives the starting number, it prints out the sequence of numbers till reaching 0. Use the "while" statement

- **do** *statement* **while (***condition***);**

```
do {
        System.out.println("Enter number (0 to end): )";
        n = Integer.parseInt( console.readLine() );
        System.out.println("You entered: " + n);
} while (n != 0);
```


- **for (***initialization***;** *condition***;** *increase***)** *statement*;
- Quiz:
  - Do the count down exercise
- Quiz:
  - Make a program that sums up numbers one to ten and prints out the result
- Quiz:
  - The user is asked for an integer, the program does the count down for the given integer. The result is printed out (one number in a separate row).

- The *break* instruction

```
for (n=10; n>0; n--) {
        System.out.println("");
        if (n==3) {
                System.out.println("countdown aborted!");
                break;
        }
}
```

- The *continue* instruction

```
for (int n=10; n>0; n--) {
      if (n==5) continue;
      System.out.println( ", ");
}
```

s4

- The selective Structure: *switch*

```
switch (expression) {
 case constant1:
   block of instructions 1
   break;
 case constant2:
   block of instructions 2
   break;
 .
 .
 .
 default:
   default block of instructions
 }
```

s4

- Example

```
switch (x){
      case 1: case 2:
        System.out.print("first");
        System.out.println("case");
        break;
      case 3:
        System.out.print("second");
        System.out.println("case");
        break;
      default:
        System.out.print("default ");
        System.out.println("case");
}
```

s4

- Activity:
  - Make a program that is in an infinite loop showing the menu for a student registration system (i.e. 1. Add, 2. Delete 3. Update, 4. Exit) Use a "switch" for selecting an option. When an option is selected, it should print the selected option e.g. "Add a student". It should have an option to exit the program.

s4

- Quiz 1:
  - Make a program that prints out numbers one to 10 (each number is printed in a separate row). You should use the most appropriate loop structure.

stdout

- System.out.print
- System.out.println
- Allows for using the operator "+" for concatenation
  - When applied to a non-string value, the value is converted to a string

```
int x = 5;
System.out.println("x = " + x);
```

- System.out.printf
- Introduced in Java 5
- Useful to give format
- Example:

```
System.out.printf("%s\n%s\n", "Welcome to", "Java Programming");

System.out.printf("x = %d  y = %d\n",  x, y);
```

conversion specifying how to display the argument:
'd': decimal integer
'o': octal integer
'x': hexadecimal integer

'f': decimal notation **for float**
'g': scientific notation (with an exponent) **for float**
'a': hexadecimal with an exponent **for float**

'c': **for** a character
's': **for** a string.
'b': **for** a **boolean** value, so its output is "true" or "false".

argument_index: "1$" refers to the first argument,
"2$" refers to the second argument

width: Specifies the field width for outputting the argument and represents the **minimum number of characters** to be written to the output.

precision: used to restrict the output depending on the conversion.
It specifies the number of **digits of precision** when outputting floating-point values.

- Width

```
int x = 5;
int y = 10;

System.out.printf("x = %3d  y = %3d\n", x, y);
```

- Precision

```
double x = 125.5468;
double y = 24.12893;

System.out.printf("x = %3.1f  y = %3.2f\n", x, y);
```

- Argument index

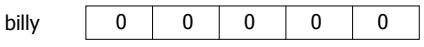
```
int x = 5;
int y = 10;

System.out.printf("x = %2$3d  y = %1$3d\n", x, y);
```

- Activity
  - Make a program that prints on the screen the ascii code (upto 127 decimal)
  - Dec Oct Hex Char

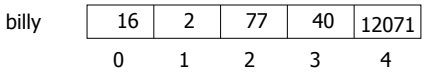
Uni-dimensional arrays

- Arrays are a series of elements (variables) of the same type placed consecutively in memory that can be individually referenced by adding an index to a unique name

- *type name* [] = new type[n];
- Example:
  - int billy [] = new int[5];

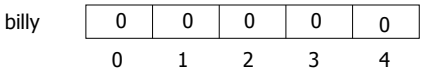


- int billy [] = { 16, 2, 77, 40, 12071 };



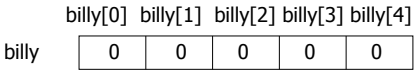
s6

- int billy[] = new int[5];



s6

- Access to the values
  - name[index]



s6

- to store the value 75 in the third *element*
  - billy[2] = 75;
- To obtain the third value
  - a = billy[2];

s6

- In Java it is perfectly valid to exceed the valid range of indexes for an Array
  - since they do not cause compilation errors
- In this case an ArrayIndexOutOfBoundsExecption is raised when running the program

s6

- Valid operations
  - b = billy [a+2];
    billy[billy[a]] = billy[2] + 5;

s6

- Traditional for

```
int total = 0;
for(int x=0;x<5;x++)
  total += billy[x];
```

- For-each loop

```
int total = 0;
for(int number : billy)
  total += number;
```

- For-each loop
  - Simplifies code
  - Can only be used for reading elements
  - The array cannot be modified