Basic Framework of a Java Program

```java
public class ClassName {
    // Introductory comment
    public static void main(String[] args) {
        variable declarations
        statements
    }
}
```
The Average of Three Values

```java
public class AverageThree {
    public static void main(String[] args) {
        int average = (2 + 4 + 6) / 3;
        System.out.println("The average of " + 2
                         + ", " + 4 + " and " + 6
                         + " = " + average);
    }
}
```

Example: Asking the user for name & age

- Available input – name and age
- Required output – a message asking the user if (s)he is $x$ years old
- Initial algorithm
  1. Ask the user for name and age
  2. Print message
Refining the algorithm for asking name and age

1. Ask the user for name and age
2. Print message

1.1 Ask the user for name
1.2 Ask the user for age

1.1.1 Prompt the user for name
1.1.2 Read the name
1.2.1 Prompt the user for age
1.2.2 Read the age
1.1.1 Prompt the user for name
1.1.2 Read the name
1.2.1 Prompt the user for age
1.2.2 Read the age
2. Print message

```java
System.out.println("What\'s your name\t?");
name = keyb.next();

System.out.println("How old are you\t?");
age = keyb.nextInt();

System.out.println("Are you really " + age + ", " + name + "?");
```
import java.util.Scanner;

public class Age {
    // Ask the user for his/her name and age
    public static void main(String[] args) {
        Scanner keyb = new Scanner(System.in);
        String name = new String();
        int age;

        // Get the name
        System.out.println("What's your name?\t");
        name = keyb.next();

        // Get the age
        System.out.println("How old are you?\t");
        age = keyb.nextInt();

        // Print the message
        System.out.println("Are you really " + age + ", " + name + "?\t");
    }
}
Improving our program

- We’ll change the program to find out which of two people is older.
- Available input – names and ages for both people
- Required output – a message stating who is older.
- Initial algorithm
  1. Ask the user for name and age of person 1
  2. Ask the user for name and age of person 2
  3. Print message

```java
System.out.println(“What’s your name	?”);
name1 = keyb.next();
System.out.println(“How old are you	?”);
age1 = keyb.nextInt();
System.out.println(“What’s your name	?”);
name2 = keyb.next();
System.out.println(“How old are you	?”);
age2 = keyb.nextInt();
```
System.out.println("What\'s your name\t?");
name1 = keyb.next();
System.out.println("How old are you\t?");
age1 = keyb.nextInt();

System.out.println("What\'s your name\t?");
name2 = keyb.next();
System.out.println("How old are you\t?");
age2 = keyb.nextInt();

3. Print message

3. if person1 is older
3.1 print person 1 is older
3.2 else if person 2 is older
3.2.1 print person2 is older
3.3 else print they\'re the same age
System.out.println("How old are you	?");
inputLine = keyboard.readLine();
age1 = Integer.parseInt(inputLine);

3. if person1 is older
3.1 print person 1 is older
3.2 else print person2 is older

if (age1 > age2)
    System.out.println(name1 + " is older than " + name2);
else if (age2 > age1)
    System.out.println(name2 + " is older than " + name1);
else
    System.out.println(name1 + " and " + name2 + " are the same age");
/ Get the name and age for person 2
System.out.println("What\'s your name\t?");
name2 = keyb.next();
System.out.println("How old are you\t");
age2 = keyb.nextInt();

// Print the message
if (age1 > age2)
    System.out.println(name1
    + " is older than " + name2);
else if (age2 > age1)
    System.out.println(name2
    + " is older than " + name1);
else
    System.out.println(name1 + " and "
    + name2 + " are the same age");
}
Iteration

• There are three different ways of writing loops in Java:
  – while loop
  – do..while loop
  – for loop

while loop

while loops are generally conditional loops and the syntax is:

```java
while (condition)
{
statement(s)
}
```

or

```java
while (condition) {
statement;
}
```
**do-while** loops

**while** loops have their test at the beginning of the loop – they might not even be executed once.

**do-while** loops have their test at the end, so they are guaranteed to be executed at least once.

The syntax is:

```java
do {
    statement(s)
} while (condition);
```

---

**Age3.java**

```java
import java.util.Scanner;

public class Age3 {
    final int namelen = 25;

    // main() - Print name and age of user
    public static void main(String[] args) {
        Scanner keyb = new Scanner(System.in);
        String name = new String(), inputLine = new String();
        int answer;
        age;

        // Get the inputs
        System.out.println("What\'s your name\t?");
        name = keyb.nextLine();
        System.out.println("How old are you\t?");
        age = keyb.nextInt();
    }
}
```
// Respond to the user
do {
    System.out.println("Are you really "+ age + ", "+ name + "?");
    inputLine = keyb.next();
    answer = inputLine.charAt(0);
} while (answer != 'y' && answer != 'n');

if (answer == 'n')
    System.out.println("I didn't think so.");
**for and while loops**

for loops can be rewritten using while loops, e.g.,

```java
for (i = 0; i < 8; i++)
    System.out.println(i);
```

is equivalent to:

```java
i = 0;
while (i < 8) {
    System.out.println(i);
    i++;
}
```

celsius1.java

```java
public class Celsius {
    // celsius - Converting from celsius to fahrenheit
    public static void main(String[] args) {
        int celsius;
        double fahr;

        System.out.println("celsius\tfahrenheit");
        for (celsius = 0; celsius < 100; celsius++) {
            fahr = celsius * 9 / 5 + 32;
            System.out.println(celsius + "\t" + fahr);
        }
    }
}
```
Data Types and Arithmetic

• When an operation is performed on two operands of the same type, the result is of that data type.
• When the operation is performed on two different data types, the result is usually the same as the data type that is higher on the hierarchy.

Celsius 1’s output

<table>
<thead>
<tr>
<th>Value</th>
<th>Temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>32.0</td>
</tr>
<tr>
<td>1</td>
<td>33.0</td>
</tr>
<tr>
<td>2</td>
<td>35.0</td>
</tr>
<tr>
<td>3</td>
<td>37.0</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
</tr>
<tr>
<td>97</td>
<td>206.0</td>
</tr>
<tr>
<td>98</td>
<td>208.0</td>
</tr>
<tr>
<td>99</td>
<td>210.0</td>
</tr>
</tbody>
</table>

Dividing 9 / 5 produces a value of 1; thus our answers are wrong.
celsius2.java

```java
public class Celsius2 {

    // celsius - A conversion table from celsius
    //    to fahrenheit
    public static void main(String[] args) {
        int celsius;
        double fahr;

        System.out.println("celsius\tfahrenheit");
        for (celsius = 0; celsius < 100; celsius += 5) {
            fahr = celsius * 9.0 / 5.0 + 32;
            System.out.printf("%d\t%7.3f\n", celsius, fahr);
        }
    }
}
```

Introducing Functions

- Functions allow us to plan and write modular programs with more easily reusable code:
- The general syntax for a function:

```java
public static DataType funcName (parameters) {
    declarations, if any
    statement(s)
}
```
**void functions**

- Functions do not have to return a value to the main program or to the function that called it. These are called void functions.
- void functions begin with the word void to indicate that no result is returned:

```java
public static void funcname(parameters) {
    declarations, if any
    statement(s)
}
```

---

**celsius3.java**

```java
public class Celsius3 {

    // celsius - A conversion table from celsius
    //    to fahrenheit
    public static void main(String[] args) {
        int celsius;
        double fahr;

        System.out.println("Celsius\tFahrenheit");
        for (celsius = 0; celsius < 100; celsius += 5) {
            fahr = fahrenheit(celsius);
            printTemp(celsius, fahr);
        }
    }
}
```
public static double fahrenheit(int celsius) {
    return celsius * 9.0 / 5.0 + 32;
}

public static void printTemp(int celsius, double fahr) {
    System.out.printf("%d\t%7.3f\n", celsius, fahr);
}


Nested ifs

- Sometimes, there are more than two clear alternatives:
  
  ```java
  if (x > 0)
      System.out.println("x is positive.");
  else if (x < 0)
      // x must be <= 0
      System.out.println("x is negative.");
  else
      // z must be 0
      System.out.println("x is zero.");
  ```
A simple menu

... ... ...
char choice;

System.out.println("Pick a transaction:");
System.out.println("(D)eposit");
System.out.println("(W)ithdrawal");
System.out.println("(B)alance query");
System.out.print("Enter an option (d/w/b)?");
choice = (char) System.in.read();

if (Character.toLowerCase(choice) == 'd')
    processDeposit();
else if (Character.toLowerCase(choice) == 'w')
    processWithdrawal();
else if (Character.toLowerCase(choice) == 'b')
    getBalance();
else
    error();

... ... ...
switch

- A `switch` statement allows you to jump to a particular part of a block depending on the value of an integer expression.

- The form is:

```java
switch(expression) {
    case value1: statement(s);
    ... ...
    default: statement(s);
}
```

Our menu rewritten

```java
System.out.print("Enter an option (d/w/b)?");
choice = (char) System.in.read();
switch(choice) {
    case 'd': processDeposit();
              break;
    case 'w': processWithdrawal();
              break;
    case 'b': getBalance();
              break;
    default: error();
              // If choice is not 1, 2, or 3
}
```
break

- **break** will force the program to leave the innermost loop or **switch** statement.
- Example:
  ```
  switch(choice) {
    case 'd': processDeposit(); break;
    case 'w': processWithdrawal();
    case 'b': getBalance(); break;
  }
  ```
  In this case, choosing ‘w’ will process a withdrawal **AND** get the balance.

break and loops

```cpp
for (i = 0; i < 5; i++)
  if (x[i] < 0)
    break; // x[i] is the first negative value

for (i = 0; i < 5; i++)
  for (j = 0; j < 6; j++)
    if (x[5*i+j] < 0)
      break; // Ends the inner loop
```
Conditional Expressions

• Let’s find the larger of two values:
  \[
  \text{if } (a \ > \ b) \\
  \quad \text{max} = a; \\
  \text{else} \\
  \quad \text{max} = b;
  \]

• Using a conditional expression, we can rewrite this as:
  \[
  \text{max} = a > b? \ a \ : \ b;
  \]

Syntax for Conditional Expressions

• The syntax for conditional expressions is:
  \[
  \text{expression1} \ ? \ \text{expression2} \ : \ \text{expression3};
  \]

• If expression 1 is nonzero, the whole expression is equal to expression 2; otherwise, it is equal to expression 3.
continue

- continue causes the program to go the next iteration of the loop:

```java
for (i = 0;  i < 20; i++) {
    System.out.print(i + " ");
    if (i % 4 != 3)
        continue;
    System.out.print("\n");
}
```

- The example prints 0 through 19, four numbers to each line.