Construction of Methods

The cell or object has behavior and is able to communicate with other cells via cell receptors or functions/methods as known in OO programming languages. The method carries out an action, which makes changes to the cell organelles (instance fields) or object contents therefore there exists a relationship between a cell and a receptor meaning modifying the contents of the cell is done through the receptor and can be expressed as **cell.receptor** or in OO programming as **object.method**.

In order to make a change to the object a method is required, that is method **l** from OOC. It has an action and can be identified using a calculus structure **f(x)**. This calculus structure can be transformed into Java code as maybe a deposit method in the following way **deposit()**. Looking closely at the method it contains the same structure as the calculus function **f(x)**. A visual method summary is shown below in Fig. 7.

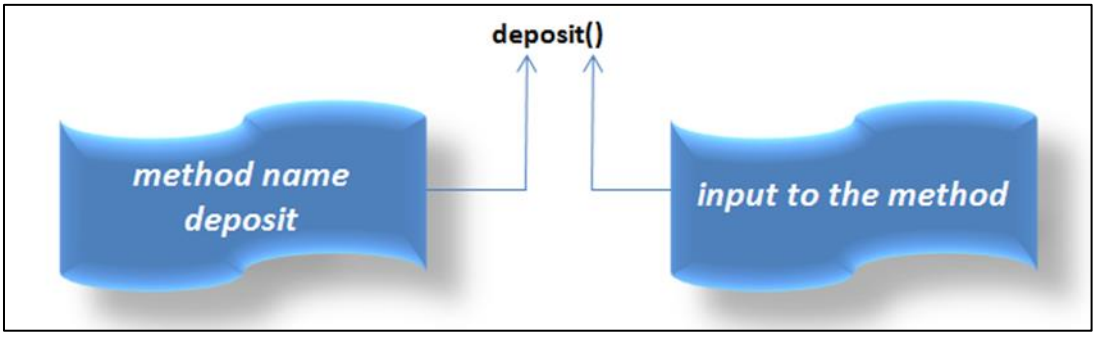

Fig. 7

Any method can be created with this structure using Java code. Again, deposit() is the equivalent method l in OOC shown in the table below.

Java Code, OOL	Procedural Language, OOC
deposit()	**method l**

There is an association between the object and the method that is **object.method** or **o.l**. This association can be transitioned into Java code as **anAccount.deposit()** where **anAccount** is the name of the object and **deposit()** is the name of the method. A visual object dot method summary is shown in the table below.

Java Code, OOL	Procedural Language, OOC
anAccount.deposit()	**o.l**

The expression **ς(x)b** can be transformed into Java code in the following way:

```
public void deposit (double anAmount) {

        double tempBalance = anAmount + currentBalance;
        currentBalance = tempBalance;
}
```

The **deposit(double anAmount)** means the following: **deposit** is the method and is equivalent to the **ς** in OOC and **double anAmount** is the explicit parameter or the input to the method meaning **double** is the data type and the **anAmount** is the name of the data type equivalent to the **x** in OOC. The body of the method is the code between the parentheses {…} also known as a block and is equivalent to the body **b** in OOC.

A visual method with input summary is shown in the table below.

Java Code, OOL	Procedural Language, OOC
deposit(double anAmount)	**ç(x)**
double tempBalance = anAmount + CurrentBalance; **currentBalance = tempBalance;**	**body *b***

The method **deposit()** is a mutator method, because it contains the keyword **void**, which specifically states that it does not return a value and it's a **public** method, since it has an access specifier **public** meaning it can be used by other classes.

Another type of a method is an accessor method meaning it does not change the contents of the object but instead displays the contents of the object.

```
public double getCurrentBalance(){

        return currentBalance;

}
```

An accessor method always has a return type in this case it is a double. Furthermore, the method is created by using the get command with a name of a method that is **getCurrentBalance()**. The **return** statement instructs the method to terminate and return the output of the method meaning the contents of the object, because following the return statement is the instance field **currentBalance** and that is where the object stores its contents.

Self-Check Questions for The Construction of Methods Sub-Section

1. What does a method carry out?
2. What is a connection operator that is used between and object and a method?
3. What is an explicit parameter?