# Object thinking in paradigms of programming

Stepan Hubalovsky and Ondrej Korinek

***Abstract*—**Thinking people use in their daily activities and thinking helps them to solve the situations, problems of everyday life. To solve the problem we are using the information we know. The information are stored in memory. When solving new situations we are used already known information. The already know information has to be combine to successfully solve a new problem.

The same situation programmers has to deal with during the design of new programs. Programmer has to combine various processes or parts of code that has ever used.

Programming is linked to several programming paradigms. Paradigms bring new programming options and replace the old one. A typical example is the replacement of a structured paradigm by object paradigm.

The paper analyzes two basic options for teaching of programming - object-oriented paradigm versus structured & object-oriented paradigm with respect to object-oriented thinking. The paper characterized object thinking as well as it shows types of examples for studying of thinking of two groups – object oriented and structured & object oriented.

The research was carried out in the Faculty of Science, University of Hradec Kralove.

***Keywords*—**Thinking, object thinking, object oriented programming, structured programming.

## I. Introduction

MANY courses of programming for students – beginners begin with structured programming and after handling it the basic approach is followed by learning of object-oriented programming (OOP hereinafter). Learning by means of objects usually follows up at the very end of the course of programing. Difficulties of learning of this methods may be several.

One of the problems is that students do not have enough time to understand object-oriented thinking. Other problem is that the students understand object oriented programming only partly, they are still use and think in terms of structured programming and do not use benefit of object oriented programming [1]. It takes at least 6-18 months [2] then programmer of structured programming completely is reoriented towards OOP.

On the other hand the requirements of market are clear: programmer who does not handle the object oriented paradigm has on the labor market far less exercise than the one that handles it.

Why doesn't learning of programming focus entirely to object oriented paradigm? Structured programming has not so much use.

There may be several reasons:

- Teachers are experienced by a structured approach and do not like to change their ingrained habits.
- The problem is also when change the structured programming paradigm to object oriented programming. Some teachers start with object oriented paradigm after the third of course, others in the middle, others in the end of the course and some of them object oriented programming do not teach all.
- Some argue that structured programming uses algorithmic thinking which is used in everyday life.

Based on authors experience not only algorithm thinking, but also object thinking is important in everyday life.

## II. Thinking

Definition thinking is difficult. There are several ways to define thinking.

"Thinking is based on the relationships between concepts" [3]. The concept we have stored in long-term memory. If we solve the problem, we use the information that we already know. We can then oriented in unfamiliar surroundings [3].

The thinking can be divided to the different forms [4]:

- deduction;
- induction;
- sorting;
- comparison;
- analysis;
- synthesis;
- generalization;
- abstraction.

### *A. Object Thinking*

Although the definition of the thinking is difficult, the definition of object oriented thinking satisfies the above definition.

When programming the programmer is using a variety of combinations of terms related to previous practical experience and theoretical knowledge of programmer. Tasks that

Stepan Hubalovsky is assoc. prof. and supervisor of Ondrej Korinek. He works at University of Hradec Kralove, Department of informatics, Faculty of Science, Hradec Kralove 500 38, Rokitanskeho 62, Czech republic, stepan.hubalovsky@uhk.cz.

Ondrej Korinek is Ph.D. student at University of Hradec Kralove, Department of informatics, Faculty of Science, Hradec Kralove 500 38, Rokitanskeho 62, Czech republic, ondrej.korinek@uhk.cz.

ISBN: 978-1-61804-322-1

programmer solves, are also problematic in most cases. Among the forms of learning that are associated with the object of thinking, will include: deduction, induction, sorting, abstraction, analysis and synthesis.

## III. Basic Concepts and Paradigm of the Programming

Algorithm can be represented in several ways - in the form of flowcharts, pseudocodes or structure-grams. It always depends on the teacher, which type of algorithm representation prefers. Algorithm development is the basis of programming [5].

The first aspect that influences learning of algorithms and programming is influenced by the form of performed teaching:

- structured form, that teaching is divided into learning algorithms, structured programming and object-oriented programming in the end;
- object oriented form, i.e. from the beginning of the instruction focuses on object-oriented programming, with the principles of the algorithms are part of this instruction.

The paper analyzes basic options for teaching of programming based on object-oriented programming with respect to object-oriented thinking, so let's describe basic paradigm of object oriented programming first.

The basic paradigm of object oriented programming (OOP) is to model on the computer the real-world situation – see e.g. [6], [7], [8].

The OOP applications are developed based on already created components. The basic terms of OOP are object, event abstraction, encapsulation, inheritance and polymorphism. From a system approach point of view the objects can be understood as open sub-system of whole application. Every object - the subsystem is a complete system - consisting of elements (a list of properties, event handlers), communicates with its environment through inputs (events, parameters) and outputs (methods and parameters).

Despite the different representation of algorithm and different paradigm of programming students should improve object thinking.

### A. Concepts of Object-oriented Paradigm in Object Thinking

Object-oriented thinking is important for proper design of classes. The classes should contain methods with the appropriate parameters actually relate to the class.

Object-oriented thinking is used in connection with other basic concepts of Object oriented programming. The programmer has to analyze whose data or methods will be available for other outside objects and that data or methods will be hidden - the principle of encapsulation. Design of the program has to fulfill the principle that surrounding of the program knows as little about the object. Programmer has to consider about interface of the object [9], how it should be programmed and what should be implemented. According to [6] the term interface should precede inheritance.

The advantage in design of object-oriented program is the knowledge of formulas, such as in tasks in physics or mathematics. Using of correct formula, input of the relevant values, expressing of the resulting values is the basic mental operation for a successful solution of the problem. Physical or mathematical formulas have analogy in object-oriented programming in the form of design patterns. [10]

According to [10] the students should become acquainted with form of design pattern as soon as possible, because it is part of object oriented thinking.

## IV. Research of Methodologies of Learning in the Subject Programming

### A. Methodology of the Research

The course of Algorithms and Data Structures learned in the first semester at the Faculty of Science, University of Hradec Kralove is followed by three courses of Programming in the programming language C#.

The research investigation was carried out in the course of programming in the academic year 2013/2014. The main goal of the research was determined the comparison of two methods of teaching of programming - object-oriented programming and structured versus object-oriented programming with respect to algorithmic thinking. Students were randomly divided evenly into two groups according to the results in course of Algorithm and Data Structures.

One group of students (***structured group of student***) followed the algorithm development by structured programming in C# programming language with functions (methods) and based on algorithmic structures. The students designed structured C# programs based on similar algorithms, the already developed in course of Algorithm and Data Structures. The learning of structured programming was followed by learning of object-oriented programming. The basic terms of object oriented programming were introduced - design and definition of classes, abstract classes, methods, constructors, encapsulation, polymorphism, interfaces and inheritance.

The second group of students (***object group of student***) began immediately after the course Algorithm and Data Structure with object oriented programming (without structured programming). In this group the concept of object oriented programming was more practiced. The concept of structured programming was omitted.

Both groups of students passed a midterm exam test with similar tasks, which consisted of theoretical and practical part. Practical (programming) part was divided into object and algorithmic part. To successfully pass the test, students had to reach in every part at least 60% of correct answer.

The research investigated the influence of the different concepts on increase of objected thinking.

### B. Credit Test

The credit test consists form some different tasks. Students

ISBN: 978-1-61804-322-1

have to correctly designed class first. They initialize one dimensional array (sequence) and private and public data items by constructor. Algorithm constructions for one dimensional array create classes in methods. Methods can or cannot return values. Return value can be in the form of primitive data type, array or class.

The OOP part (definition of class, constructor, and method) of the test is evaluated separately from algorithmic part.

Students have to define some method based on algorithmic areas. They have to fulfill one task from each algorithmic area.

Separated from algorithmic structures in evaluation of the task.

**Sample of credit test:**

Create class *Sequence* for sequence operation (one-dimensional array) with the following components:

- *Constructor* - creates private data item of type of one-dimensional array of integers of a given size.
- *N* - read-only property specifying the length of the sequence.

***Filling the sequence:***

- *FillFibonaccim* - using the method input the values of Fibonacci sequence to array values (1, 1, 2, 3, 5, 8, 13, 21 ...)

***Output the sequence:***

- *WriteRow* - using the method writes a sequence in the console line (values are separated by a comma).
- *Member* - using the method returns the value of element at the position specified as an input parameter.

***Calculations and search in the sequence:***

- *Average* - calculates and returns the average value of the terms of the sequence.

***Shifting the values in the sequence:***

- *Reversion* - reverses the order of the members in the sequence (e.g. from 1,2,3,4 to 4,3,2,1).

***Inserting / removing values:***

- *RemRand* - removes a randomly selected member of the sequence

***Working with multiple sequences:***

- *SumArray* - adds this sequence with the other sequences of the same length (has to be verified) specified as input parameter of the new third sequence, which returns as a return value of type Sequence.

In the main part of the program (method *Main* of class *Program*) create an instance of the class Sequence and properly use all the methods implemented.

### *C. The Result of the Research – Object Task*

The first credit test was based on practical tasks. The algorithmic task was based on one-dimensional array. Students should propose algorithms of given problem.

The first algorithmic group of students consists of 9 students participated in the test.

The second object oriented group of students consists of 8 students participated in the test.

To determine whether the median of the result of student achieved in object part is the same for the first and second groups of students was used a nonparametric Mann-Whitney test.

Calculated P- value is P = 0.00127

with significance level $\alpha = 0.05$,

so we can reject the null hypothesis that the median of students results of the object part between the groups is the same. Between groups is statistically significant difference.

Box plots diagrams of both groups of students are in figure 1.

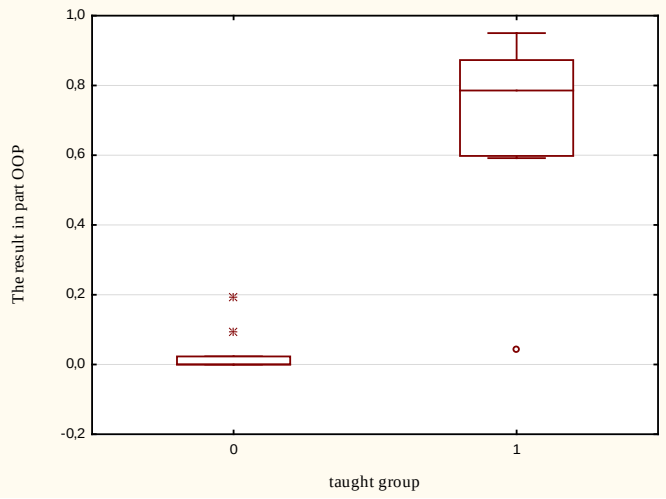


Fig. 1 Box plot diagram comparing result of algorithmic group of students with object oriented group of students in algorithmic test.

The results show that the first algorithmic group of students completely failed in object oriented thinking, no student succeeds in the test.

In the second object oriented group 6 of 8 students succeeded.

Interesting results can be also reached from analysis of the code of programs (will be published later).

The basic object oriented skills include proper design classes, creating characterizing methods associated with the class, object calls and creating specific objects.

All these skills were in the student projects examined separately.

### *D. The Result of the Research – Object Oriented Task 1*

In the first part of object oriented test was design classes, constructors, methods with primitive and object types examined.

To determine whether the median of the results correct initialization data elements using the constructor of the programs is the same for the first algorithmic group of student and second object oriented groups of students the nonparametric Mann-Whitney test was used.

Calculated P- value is P = 0.00064

with significance level $\alpha = 0.05$

Between both groups is statistically significant difference. We can reject the null hypothesis that the median of both

group of students is the same. Box plots of both groups of students are on Figure 2. The graph shows that the worst student of the second OOP group has higher score than the best student of the algorithmic group.

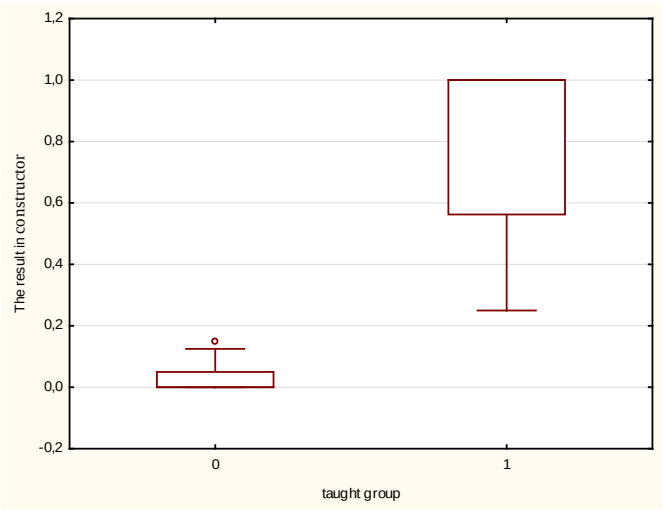


Fig. 2 Box plot diagram comparing result of algorithmic group of students with object oriented group of students in object oriented test 1 - design classes, constructors, methods with primitive and object types.

### *E. The Result of the Research – Object Oriented Task 2*

In the second part of object oriented test was create instance with specified values.

To determine whether the median of the results of the correct definition of methods with parameters of primitive and object types in the programs is the same for the first and second groups of students the nonparametric Mann-Whitney test was used.

Calculated P - value is P = 0.0033

With significance level $\alpha = 0.05$

Between both groups is statistically significant difference. We can reject the null hypothesis that the median of both group of students is the same. Box plots of both groups of students are on Figure 3.

From the graph it is clear that most student of the first algorithmic group of students didn't succeed the task. The student are not able defined methods in the class as well as defined object types.

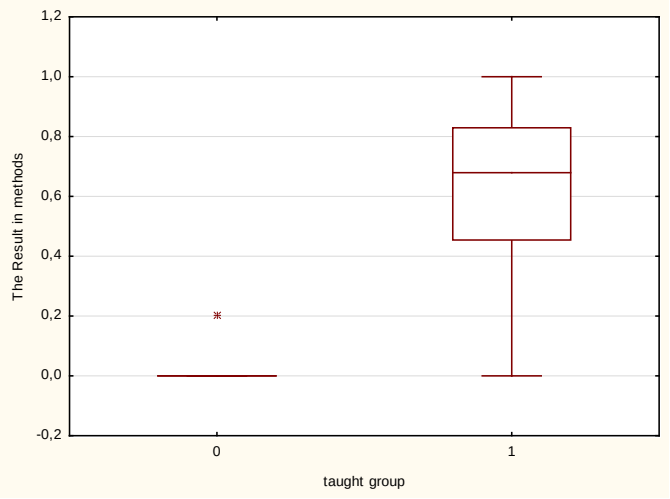


Fig. 3 Box plot diagram comparing result of algorithmic group of students with object oriented group of students in object oriented test 2 – creation of instance with specified values.

### *F. Sample of Result of Tests*

Students OOP do not have problems with initialization data elements using the constructor. They can without any problems create instance of the class in the main part of the program. The student have the greatest problems with the return value in the method. Some of them cannot return even primitive data type or object type.

Students structured follow-up and object-oriented programming have problems with the whole object parts. They cannot properly create class, constructor or method.

**Example of task:**

Create method that determine the number of members whose value equals to the value specified as input parameter and the result value returns as output value.

Example the correct solution is shown in figure 4.

```
public int Pocet(int hodnota)
{
    int pocet = 0;
    for (int i = 0; i < N; i++)
    {
        if (posloupnost[i] == hodnota)
            pocet++;
    }
    return pocet;
}
```

Fig. 4 Method – determination of the number of members – correct solution.

Example of wrong student's solution is in figure 5 (wrong settings of input and return value of the methods. The value is not returned but the value is written).

ISBN: 978-1-61804-322-1

```
public void Pocet()
{
    int y = 0;
    int  x = Convert.ToInt32(Console.ReadLine());
    for (int i = 0; i < N; i++)
    {
        if (posloupnost[i] == x) y = y + 1;
    }
    Console.WriteLine(y);
    Console.WriteLine();
}
```

Fig. 5 Method – determination of the number of members – wrong student's solution.

## V. Conclusion

The paper includes definition of object thinking and compared the two teaching methods of programming with respect to this thinking.

Based on result of our research it is clear, that no student from the first group (first structured programming than object oriented programming) succeeded in the object oriented part of the test.

On the other hand 75 % of students from the second group (only object oriented programming) succeeded the same test.

Detailed analysis of the results of the test discovered that the results of the first group was bad. The students had problems with the basic concepts of OOP in all parts of the proposal.

The causes of such failure can be several:

- shorter time spend by training of principles of object oriented programming
- different way of thinking that students understand.
- teacher's approach, because each group was taught by different teacher. To eliminate this factor, we will provide in this academic year the same research with the same teacher for both groups.

The results provide feedback based on which the learning of algorithm and programming will be modify.

## Acknowledgment

This research has been supported by: Specific research project of University of Hradec Kralove, Faculty of Education in 2015 and Specific research project of University of Hradec Kralove, Faculty of Science in 2015.

## References

[1] J. Bergin, "Fourteen Pedagogical Patterns", in *Proc. of Fifth European Conference on Pattern Languages of Programs (EuroPLoP™ 2000)*, Irsee, 2000.

[2] R. Pecinovsky, "*Learn Object Oriented Thinking and Programming*". Brno, Computer Press 2010.

[3] I. Ruisel, „*Základy psychologie inteligence*". Praha, Portál, 2000.

[4] D. L. Schacter, D. T. Gilbert, D. M. Wegner, "*Psychology*", Bedford, Worth Publishers, 2011.

[5] S. Hubalovsky, "Research of Methods of a Multidisciplinary Approach in the Teaching of Algorithm Development and Programming", in *Proc. DIVAI 2012 – 9th International Scientific Conference on Distance Learning in Applied Informatics*, 2012. pp. 147 – 156.

[6] J. A. Sajaniemi, Chenglie, "Teaching Programming: Going beyond "Objects First"", 2006 in *Proc. Psychology of Programming*, [online]. Available: http://www.ppig.org/papers/18th-sajaniemi.pdf

[7] M. Ricken, *Assignments for an Objects-First Introductory Computer Science Curriculum*, 2005, [online]. Available: http://www.cs.rice.edu/~javaplt/papers/tr200504.pdf

[8] J. Bennedsen, C. Schulte, "What does objects-first mean?: An international study of teachers' perceptions of objects-first", in *Proc. of the Seventh Baltic Sea Conference on Computing Education Research*, Australian Computer Society, Inc., vol. 88, pp. 21-29, 2007. Available: http://crpit.com/confpapers/CRPITV88Bennedsen.pdf

[9] R. Pecinovsky, "Early Introduction of Inheritance Considered Harmful", in *Proc. Objects 2009*, Hradec Králové, 2009.

[10] R. Pecinovsky, "Principles of the Methodology Architecture First", in *Proc. Objekts 2012*, Praha, 2012.

**Stepan Hubalovsky** was born in Trutnov, Czech Republic in 1970, he obtained master degree in education of mathematics, physics and computer science in 1995, Ph.D. degree in theory of education in physics in 1998 both in Faculty of Mathematics and Physics, Charles University in Prague, Czech Republic and assoc. prof. degree in system engineering and informatics in 2012 in University oh Hradec Kralove, Czech Republic.
He worked 5 years as master of mathematics, physics and computer science on several secondary schools. He works as associate professor on University of Hradec Kralove from 2006. He interested in algorithm development, programming, system approach, computer simulation and modelling.
Assoc. prof. RNDr. Stepan Hubalovsky, Ph.D. is member of Union of Czech Mathematicians and Physicist.

**Ondrej Korinek** was born in 1985 in Horice, Czech Republic. He graduated from the University of Hradec Kralove, Czech Republic in 2010, where he studied Education Mathematics and Computer Science for Secondary Schools.
Since 2012, he has continued his studies of Information and Communication Technology in Education at postgraduate level. Since 2010 he has been working as an ICT teacher at VOS and SPS in Jicin, Czech Republic. He is interested in algorithms, programming methodology, database systems and modern technology.