An Intelligent Tutoring System for Learning Java Objects

S. Abu-Naser[1], A. Ahmed[1], N. Al-Masri[1], A. Deeb[2], E. Moshtaha[2] and M. Abu-Lamdy[2]

[1]Faculty of Engineering and Information technology, Al-Azhar University, Gaza, Palestine.

[2]Faculty of Information Technology, Palestine University, Gaza, Palestine.

Corresponding author: Samy S. Abu-Naser, Faculty of Engineering and Information technology, Al-Azhar University, Gaza, Palestine. Tel:(+9708)2824020 Fax:(+9708)2832907

ABSTRACT

The paper describes the design of a web based intelligent tutoring system for teaching Java objects to students to overcome the difficulties they face. The basic idea of this system is a systematic introduction into the concept of Java objects. The system presents the topic of Java objects and administers automatically generated problems for the students to solve. The system is dynamically adapted at run time to the student's individual progress. The system provides explicit support for adaptive presentation constructs. An initial evaluation study was done to investigate the effect of using the intelligent tutoring system on the performance of students enrolled in computer science III in the Faculty of Engineering and Information technology at Al-Azhar University, Gaza. The results showed a positive impact on the evaluators.

KEYWORDS:

Intelligent Tutoring System, Java Objects, Problem Generation, JO-Tutor

1. INTRODUCTION

Intelligent Tutoring Systems (ITSs) can be traced back to the early 1970s, when Carbonell tried to combine methods of Artificial Intelligence (AI) with Computer Aided Instruction (CAI)[9]. Thus, the first generation of ITSs are more or less a kind of "intelligent" CAI. Their main task is stated by Lelouche: "The basic principle of 'intelligent' CAI is that it should know the taught material"[15]. Knowledge about the taught material is embedded in the ITS in form of expert systems, that is, the expert module [3,11]. The integration of insights of cognitive science in ITSs, has led to what today is called an Intelligent Tutoring System[2]. In addition to the knowledge about the taught material, these systems have knowledge about pedagogical strategies and knowledge about the student, realized as pedagogical module and student module, respectively.

The classical ITS architecture, first described by Clancey, consists of the components: expert module, pedagogical module, student module, and user interface[10]. The naming of the components varies. Sometimes, depending on the training domain, a component for automatic generation of exercises is also part of the ITS. Whereas the ITS's constituents seem to be part of common agreement in the ITS community, the role and the functionality of each of the components varies a lot[1,3]. The reason for this can partly be seen in the different application domains. One can easily imagine that training in mathematics places different demands on ITSs than clinical medicine training. Another reason might be the realization of different learning theories in ITS. Case-based learning, described in the former section, places special demands on an ITS that are somewhat different in problem-oriented learning. These reasonable aspects often necessarily lead to heterogeneous and incomparable systems. But even in the same application domain and based on the same learning style, ITSs are often not comparable and based on a complete different interpretation of the same architecture. Moreover, regarding only the ITS architecture on a more abstract level, it becomes hard to find reasons for heterogeneous realizations at all. A mixture of content and delivery functions, which is seemingly not based on insights of research but on traditions of ITS development, can be found in ITS realizations.

In other ITSs, the expert knowledge base is a simple database without own functionality[17]. The same situation can be found in the different ways the pedagogical knowledge module is realized and embedded in the ITS. Thus, there are ITSs that consist of a set of interacting and more or less separate subsystems, and there are ITSs consisting of passive components plus a component that encapsulates the execution. Execution in this context contains the interaction with the student, the evaluation of the student's behavior and success, and the provision of contents and navigation. Thus, two perspectives on the same architecture can be found, reflecting different interpretation of the same modules:
ITS architecture consists of either separate independent subsystems or passive components with centralized execution system.
Both perspectives have their advantages and disadvantages. However, the main system's philosophy regarding the realization of the components should be made clear to provide for comparability of ITSs and reusability of ITS components. The advantage of this approach is that the central steering component might be reused in different ITSs, as it is obviously separated from the databases and the user interface. Figure 1 shows the suggested ITS architecture with the tutoring process module is sketched.

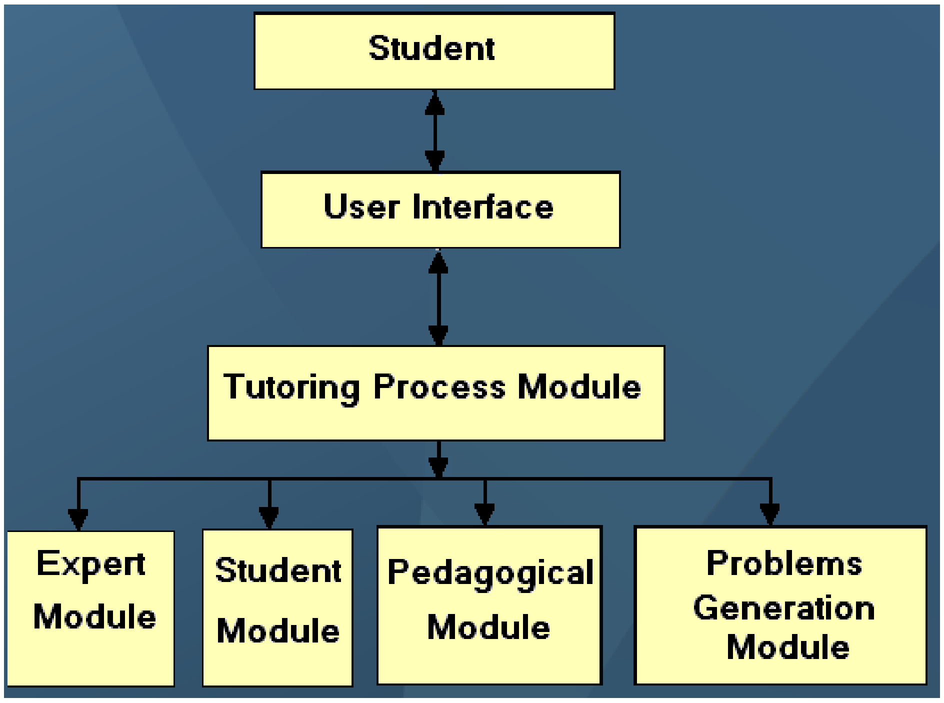


Figure 1. ITS architecture with tutoring process module

The design of the Intelligent Tutoring System for Learning Java Objects (JO-Tutor) adopted the ITS architecture with automatically generated problems module.

The objectives of the JO-Tutor are:

- To build an intelligent tutoring system for problems for which the answers might not be always quantitative

- To be able to generate unlimited number of problems automatically, thereby providing as much practice with problem-solving as the student needs.

- To build familiar interactive interfaces like OS desktops

- To have a system that is dynamically adapt at run time to the student's individual progress

2. JO-Tutor Design

Over the years a few innovative tutoring tools have been studied in an attempt to improve the quality, flexibility and cost effectiveness of teaching and learning. Kashy developed a tutor called CAPA to help students in solving Physics problems[12].

Barker have developed a tutor for homework assignment in electronic and control system discipline[6].

Bridgeman developed Interactive tutors like: PILOT and SAIL[7,8]. PILOT was for Learning and grading. PILOT is a problem generation tool for graph algorithms, while SAIL is a LaTeX-based scripting tool for problem generation.

JO-Tutor is unique with respect to the previous work in the following manner:

- The system was built to look like the most familiar interactive interfaces like OS desktops, including (icons, drag and drop features, drop down menus, and pop up windows) which are all integrated in one single window.

- The intelligent tutoring system was built for problems for which the answers might not be always quantitative [1,2].

- WebToTeach is related to JO-Tutor[5], but it administers instructor pre-prepared problems, and does not generate the problems automatically. JO-Tutor can generate unlimited number of problems automatically, thereby providing as much practice with problem-solving as the student needs.

Kashy have shown that the use of problem generation systems have increased student performance by 10% in Physics, largely due to limitless time spent on the task[13].

JO-Tutor is designed to help students learn Java Object concepts by:

1. Gradually teaching the Java Object material to the students. The system is supported with a student controlled voice narrator, which acts as a facility during learning.
2. Repeatedly solving automatically generated problems and
3. Obtaining the proper feedback.

The JO-Tutor is designed to be used as a supplement to the traditional method of teaching (Java textbook and Instructor), either during a laboratory session, after class training, or for homework assignments.

JO-Tutor has the following modules: Pedagogical Module, Problems Generation Module, Expert Module, Student Module, and Tutoring process module.

2.1 JO-Tutor Pedagogical Module Design

It has been noticed that students are having difficulties in understanding the concepts of Objects in Java. To overcome these difficulties, an Intelligent Tutoring System for teaching Java Objects called JO-Tutor have been developed to students enrolled in Computer Science III in the faculty of Engineering and Information Technology at Al-Azhar University in Gaza. JO-Tutor gradually introduces students to the concept of Java objects and automatically generates problems for the students to solve. The key sections that draws the main structure of the tutoring material[18,19,20] are:

3. **Classes and Objects**
   - Classes, Objects, Methods and Instance Variables
   - Declaring a Class with a Method
   - Instantiating an Object of a Class
   - Declaring a Method with a Parameter
   - Instance Variables, set Methods and get Methods
   - Primitive Types and Reference Types
   - Initializing Objects with Constructors
   - Access modifiers
   - Composition
   - Enumerations
   - Static Class Members
   - Final Instance Variables

4. **Inheritance**
   - Superclasses and Subclasses
   - protected Members
   - Relationship between Superclasses and Subclasses
   - Constructors in Subclasses
   - Nested Classes

5. **Polymorphism**
   - Introduction to Polymorphism and Examples
   - Demonstrating Polymorphic Behavior
   - Abstract Classes and Methods
   - Final Methods and Classes

2.3 Problems Generation Module of JO-Tutor

Limited number of problems has been recently acknowledged as a potential shortcoming of encoding a finite set of problems into a tutor [16]. A method used in literature to dynamically generate problems is by using BNF-like grammar [14]. In this method, problems are generated by randomly instantiating templates written in the grammar. Each template can be carefully designed with specific pedagogical objectives in mind.

Problem Generation Module is responsible for generating the template of code, theses templates cover the main topics of Java Object Oriented Programming which are (functions, classes, Inheritance & polymorphism). The module depends on randomly structuring of the pieces of codes which forms the templates that consist of (access modifiers types, return types, arguments data types, classes, methods, and arguments names). These structures are imported from pre defined lists of keywords.

The template is generated with a previously intended problem each time it is requested, followed by a related question and possible solutions. The questions have different styles including either asking the student to correct a Java code, write a Java code, multiple choice, or true/false. From the perspective of enhancing high availability of required tools for student during learning process, the system provide a simple editor linked with a Java compiler, to enable the user to test some pieces of Java code he wants, so that there is no need to get out of the system environment to compile Java codes.

2.4 Expert Module of JO-Tutor

Expert Module was implemented to gather the necessary information for generating the feedback[4]. The expert module is capable of solving the generated problems by parsing the template. Since the expert module can execute any code, it can generate the correct answer for a problem on its own, and determine whether the user's answer is correct/incorrect. In addition to whether the user's answer is correct/incorrect, the module can provide the student with the correct answer when it is requested. Furthermore, the module provides the student the proper feedback in response to the student's answer.

2.5 Student Module of JO-Tutor

A new student must create his own account to have a profile. The profile has information about the student such as his name, dates of login, score of each session, and learning progress during the each session. The student's score can be viewed at any time during the session as a 3D Bar chart that describes the student performance in solving problems in the following subjects: casting, classes and inheritance.

2.6 Tutoring process module of JO-Tutor

Tutoring process module works as a coordinator that controls the functionality of the whole system.

3.JO-Tutor User Interface Design

Figure 2 shows the user interface of the tutor that consists of icons and drop down menus. Once the main screen is shown up, another screen is shown to enable the user to login into the system. If the user is using the system for the first time, he should create a new username and password (See Figure 3). After the user enters the correct username and password the system gives control to the user of the main screen. From the main screen, the user can click on the clock icon for generating questions(true/false, multiple choice, and correct Java code questions (see figure 4)), the chicken icon for learning the material of Java Object where the user choose the topic to learn(see figure 5), wheel icon for help about the system, notes icon for personal notes, question mark icon shows the work team, editor icon for typing and compiling java codes(see figure 6), and Pie chart icon for showing the user current scores in casting, classes and inheritance(see figure 7). Furthermore, the user can use the pop up menu to select any of the functions of the icons.

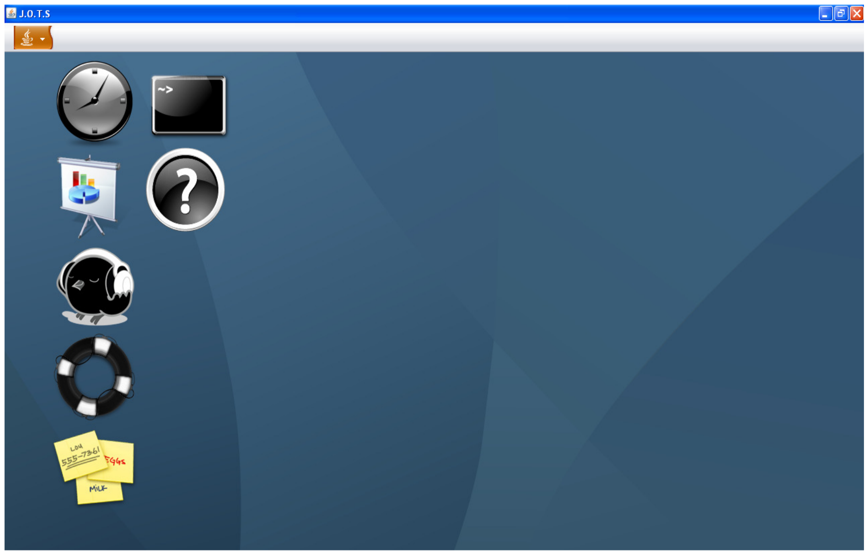


Figure 2: The main user interface

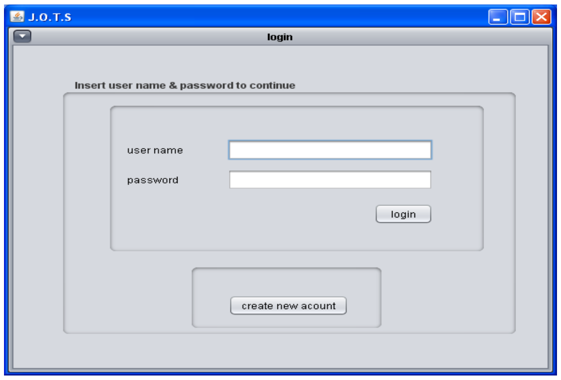
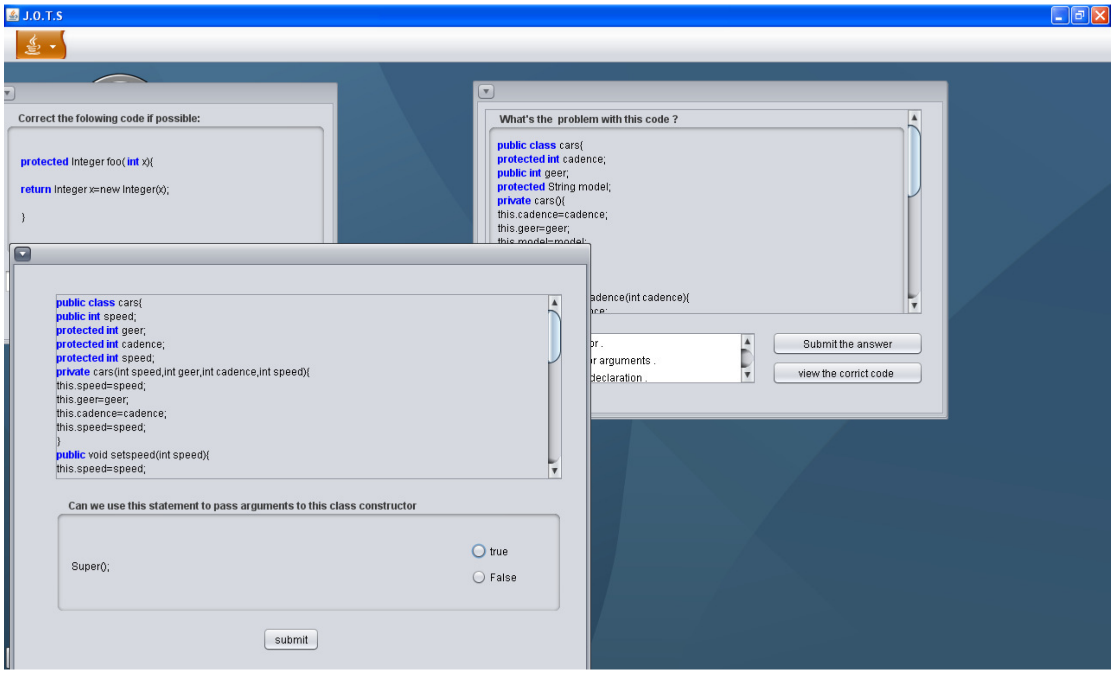


Figure 3: The login screen



Correct the folowing code if possible:

```
protected Integer foo( int x){

return Integer x=new Integer(x);

}
```

What's the problem with this code ?

```
public class cars{
protected int cadence;
public int geer;
protected String model;
private cars(){
this.cadence=cadence;
this.geer=geer;
this.model=model;
```

```
adence(int cadence){
ce;
```

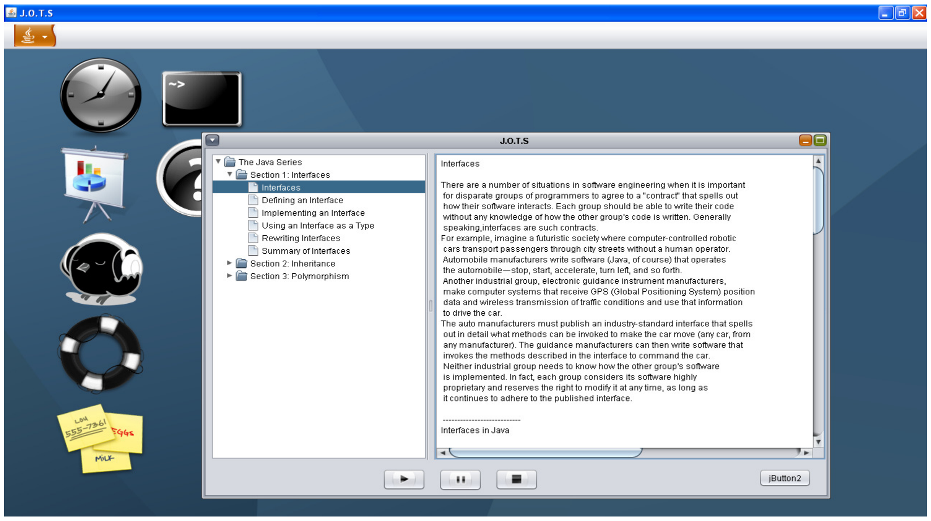
```
or .
r arguments .
declaration .
```

```
public class cars{
public int speed;
protected int geer;
protected int cadence;
protected int speed;
private cars(int speed,int geer,int cadence,int speed){
this.speed=speed;
this.geer=geer;
this.cadence=cadence;
this.speed=speed;
}
public void setspeed(int speed){
this.speed=speed;
```

Can we use this statement to pass arguments to this class constructor

```
Super();
```

Figure 4: Different types of questions

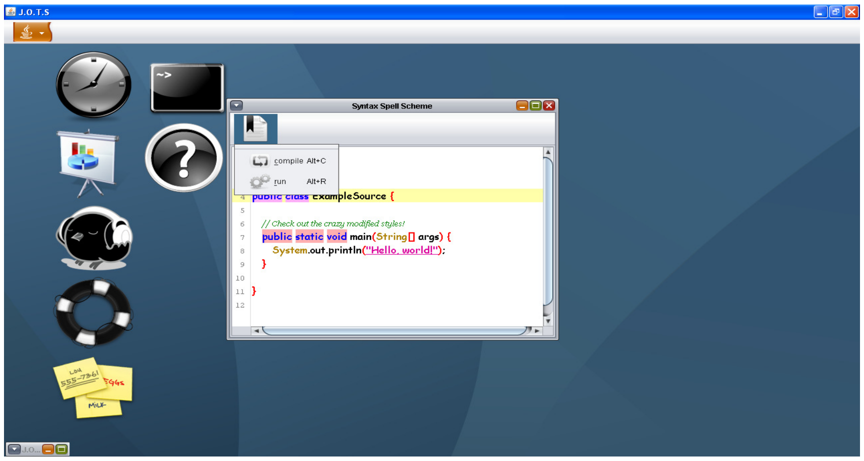


Interfaces

There are a number of situations in software engineering when it is important
for disparate groups of programmers to agree to a "contract" that spells out
how their software interacts. Each group should be able to write their code
without any knowledge of how the other group's code is written. Generally
speaking,interfaces are such contracts.
For example, imagine a futuristic society where computer-controlled robotic
cars transport passengers through city streets without a human operator.
Automobile manufacturers write software (Java, of course) that operates
the automobile—stop, start, accelerate, turn left, and so forth.
Another industrial group, electronic guidance instrument manufacturers,
make computer systems that receive GPS (Global Positioning System) position
data and wireless transmission of traffic conditions and use that information
to drive the car.
The auto manufacturers must publish an industry-standard interface that spells
out in detail what methods can be invoked to make the car move (any car, from
any manufacturer). The guidance manufacturers can then write software that
invokes the methods described in the interface to command the car.
Neither industrial group needs to know how the other group's software
is implemented. In fact, each group considers its software highly
proprietary and reserves the right to modify it at any time, as long as
it continues to adhere to the published interface.

Interfaces in Java

Figure 5: Learning material of Java Object



```
public class ExampleSource {

  // Check out the crazy modified styles!
  public static void main(String[] args) {
    System.out.println("Hello, world!");
  }

}
```

Figure 6: Editor and compiler menu

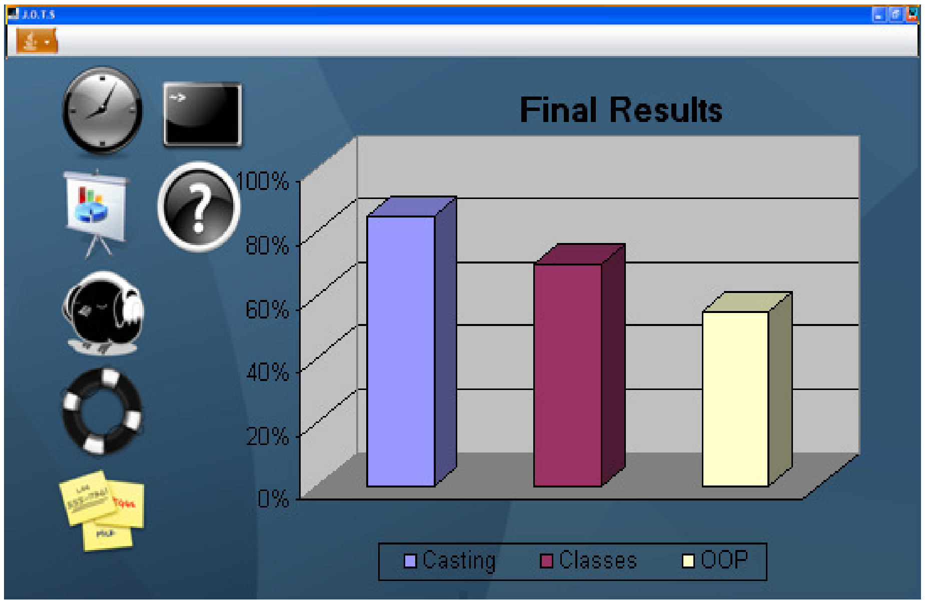


Figure 7: Final Results

4.Evaluation of the JO-Tutor

An initial evaluation of the JO-tutor was carried out by the lecturers and their students who enrolled in Computer Science III (advanced Java) during the Fall semester of 2010/2011 in Faculty of Engineering and Information Technology at Al Azhar University-Gaza. A questioner consisting of the items in table 1 was filled out by each evaluator (Lecturers and Students). A group of 3 lecturers and 20 students participated in the evaluation of the system. Table 1 shows the overall rating of the lecturers and students who evaluated the system.

Table 1: Shows the rating of the of the JO-Tutor by lecturers and students

Item #	Item	Rating %	
		Lecturers	Students
1	The Quality of the JO-Tour Design	88%	92%
2	The importance of the topic covered (advanced Java)	94%	98%
3	Would you benefit from using the JO-Tutor?	86%	98%
4	Do you recommend using JO-Tutor for Computer Science III (advanced Java) as a supportive tool?	100%	100%
5	Would you like to see similar tutoring system in other courses?	100%	100%

Form the summary of Table 1, the evaluation of the JO-Tutor showed a positive impact on the evaluators (Lecturers and students). Furthermore, they recommend that similar systems for other courses to be implemented.

5. CONCLUSIONS AND FUTURE WORKS

The design of an Intelligent Tutoring System called JO-Tutor was described in this paper. JO-Tutor was designed for teaching Java objects to students to overcome their difficulties. JO-Tutor presents the topic of Java objects to the student and administers automatically generated problems for him to solve. JO-Tutor is dynamically adapted at run time to the student's individual progress. An initial evaluation of JO-Tutor was carried out by the lecturers and students taken the advanced Java course in the faculty of Engineering and Information Technology at Al Azhar University in Gaza. The outcome of the evaluation was positive and suggested that other intelligent tutoring systems be designed for other courses. We recommend a comprehensive evaluation of the system to be carried out next time the course is offered.

REFERENCES

[1] Abu-Naser, S**., 2009.** Evaluating the effectiveness of the CPP-tutor, an intelligent tutoring system for students learning to program in C++. J. Applied Sci. Res., 5: 109 -114

[2] Abu-Naser, S**., 2008.** Developing an intelligent tutoring system for students learning to program in C++. Inf. Technol. J., 7: 1055 –1060

[3] Amalathas, S., A. Mitrovic, R. Saravanan and D. Evison, **2010**. Developing an intelligent tutoring system for palm oil in ASPIRE. Proceedings of the 18th International Conference on Computers in Education, Nov. 29-Dec. 3, Asia-Pacific Society for Computers in Education, Putrajaya, Malaysia, pp: 101-103

[4] Anderson, J.R., A.T. Corbett, K.R. Koedinger and R. Pelletier, **1995**. Cognitive tutors: Lessons learned. J. Learn. Sci., 4: 167-207.

[5] Arnow, D. and O. Barshay, **1999.** WebToTeach: An interactive focused programming exercise system. Proc. Annu. Frontiers Educ. Conf., 1: 12A9/39 - 12A9/44

[6] Barker, D.S., **1997.** CHARLIE: A computer-managed homework, assignment and response, learning and instruction environment. Proc. Annu. Frontiers Educ. Conf. Teach. Learn. Era Change, 3: 1503 - 1509

[7] Bridgeman, S., M.T. Goodrich, S.G. Kobourov and R. Tamassia, **2000**. PILOT: An interactive tool for learning and grading. Proc. SIGCSE Tech. Symp. Comput. Sci. Educ., 32: 139-143.

[8] Bridgeman, S., M.T. Goodrich, S.G. Kobourov and R. Tamassia, **2001**. SAIL: A system for generating, archiving and retrieving specialized assignments using LaTeX. Proceedings of the 31st SIGCSE Technical Symposium, **Mar. 7-12**, Austin, TX, pp: 300-304

[9] Carbonell, J., **1970**. AI in CAI: An artificial-intelligence approach to computer-assisted instruction. IEEE Trans. Man Mach. Syst., 11: 190 - 202

[10] Clancey, W., **1984**. Methodology for Building an Intelligent Tutoring System. In: Methods and Tactics in Cognitive Sciences, Kintsch, W., J.R. Miller and P.G. Polson (Eds.). Lawrence Erlbaum Associates, Hillsdale, New Jersey, London, pp: 51-84

[11] Fournier-Viger, P., R. Nkambou and E. Mephu, **2010**. Building Intelligent Tutoring Systems for Ill-Defined Domains. In: Advances in Intelligent Tutoring Systems, Nkambou, R., R. Mizoguchi, and J. Bourdeau (Eds.). Springer-Verlag, Berlin, Heidelberg, pp: 81-101.

[12] Kashy, E., B.M. Sherrill, Y. Tsai, D. Thaler, D. Weinshank, M. Engelmann and D.J. Morrissey, **1993.** CAPA-An integrated computer-assisted personalized assignment system. Am. J. Phys., 61: 1124-1130.

[13] Kashy, E., M. Thoennessen, Y. Tsai, N.E. Davis and S.L. Wolfe, **1997**. Using networked tools to enhance student success rates in large classes. Proc. Annu. Frontiers Educ. Conf. Teach. Learn. Era Change, 1: 233-237

[14] Koffman, E.B. and J.M. Perry, **1976**. A module for generative CAI and concept selection. Int. J. Man Mach. Stud., 8: 397-410.

[15] Lelouche, R., **1999**. Intelligent tutoring systems from birth to now. Kunstliche Intelligenz, 13: 5-11.

[16] Martin, B. and A. Mitrovic, **2000**. Tailoring feedback by correcting student answers. Proc. Intell. Tutor. Syst., 1839: 383-392.

[17] Tang, Y., 2005. Qualitative reasoning and articulate software. Inform. Technol. J., 4: 184-188.

[18] Daniel Liang, Y., **2010**, Introduction to Java Programming, Comprehensive (8th Edition), Prentice Hall.

[19] Deitel, P. and H. Deitel, **2009**, Java How to Program: Early Objects Version (8th Edition), Prentice Hall.

[20] Reges, S. and M. Stepp, **2010**, Building Java Programs: A Back to Basics Approach (2nd Edition), Addison Wesley.