"Ss. Cyril and Methodius" University in Skopje
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Evolution of Teaching Strategies in a Software Quality and Testing University Course

Bojana Koteska & Anastas Mishev
University Ss. Cyril and Methodius
Faculty of Computer Science and Engineering, Skopje, N. Macedonia
e-mails: {bojana.koteska, anastas.mishev}@finki.ukim.mk

Outline

- Course timeline
- Course organization and learning methods
- Student evaluation
- Evolution of student results
- Preparing students for software testing industry
- Acknowledgement

Course timeline

- 2008: Mandatory/elective course in third and last studying year for different 4-year and 3-year bachelor programs since 2008
  - at Faculty of Natural Science and Informatics, Ss. Cyril and Methodious University
- 2011: moved to Faculty of Computer Science and Engineering as elective course
- 2020/2021: mandatory course in 6th semester – English and Macedonian classes
  - study program: Software engineering and information systems

Course organization

- Total 180 hours:
  - Lectures – theoretical teaching: 30 hours
  - Exercises, seminar papers, teamwork: 45 hours
  - Project Tasks: 15 hours
  - Independent Learning Tasks: 15 hours
  - Home learning: 75 hours
- 12 lecture weeks and 2 weeks for exams
- Weekly: 2 lecture hours + 1 exercise hour + 2 laboratory exercises hours for presentation of homework tasks
- 8 homework tasks + 1 group project + 1 seminar work (optional)

Learning methods

- Presentations
- Interactive lectures
- Exercises (using equipment and software packages)
- Teamwork
- Case studies
- Invited guest lecturers
- Independent preparation and defense of a project assignment and seminar work.

Lectures

- Goals:
  - understand the need for software testing
  - different techniques of software testing
  - Learn about verification, and validation
  - Use the knowledge in practice: test real projects
- Books:
  - 2008 - 2016: First edition of “Introduction to Software Testing” by Paul Ammann and Jeff Offutt
  - 2016 – now: Second edition of “Introduction to Software Testing” by Paul Ammann and Jeff Offutt

Lecture topics

- Why Do We Test Software?
- Model-Driven Test Design
- Putting Testing First
- Criteria-Based Test Design
- Input Space Partitioning
- Graph Coverage
- Logic Coverage
- Syntax-based Testing

Exercises

- Before: “theoretical” exercises (paper-based)
- Now: computer-based testing tasks of real software programs using the latest software testing packages and frameworks
  - Unit testing: Junit 5
  - Selenium (Web automation testing)
  - Mockito (Mock objects and testing)
  - Pitclipse (Mutation Testing)
  - Graph Coverage (source code)
  - Input Space Partitioning (source code)
  - Logic Coverage (source code)

Practical projects and seminar work

- Before: project assignments were same for all students (mathematical tasks or testing the same simple software system)
- Now: each team have to choose different real software system for testing
- Seminar work: Make a research about some tool (technique) for testing and demonstrate it practically

Student evaluation

- Before: more points for midterm exams
- Now: the accent is put on the practical project and homework
  - 8 homework (100 points)
  - Seminar work (20 points) – optional
  - Practical project (100 points)
  - 2 midterm exams (150 points)

  - Total: 350 points

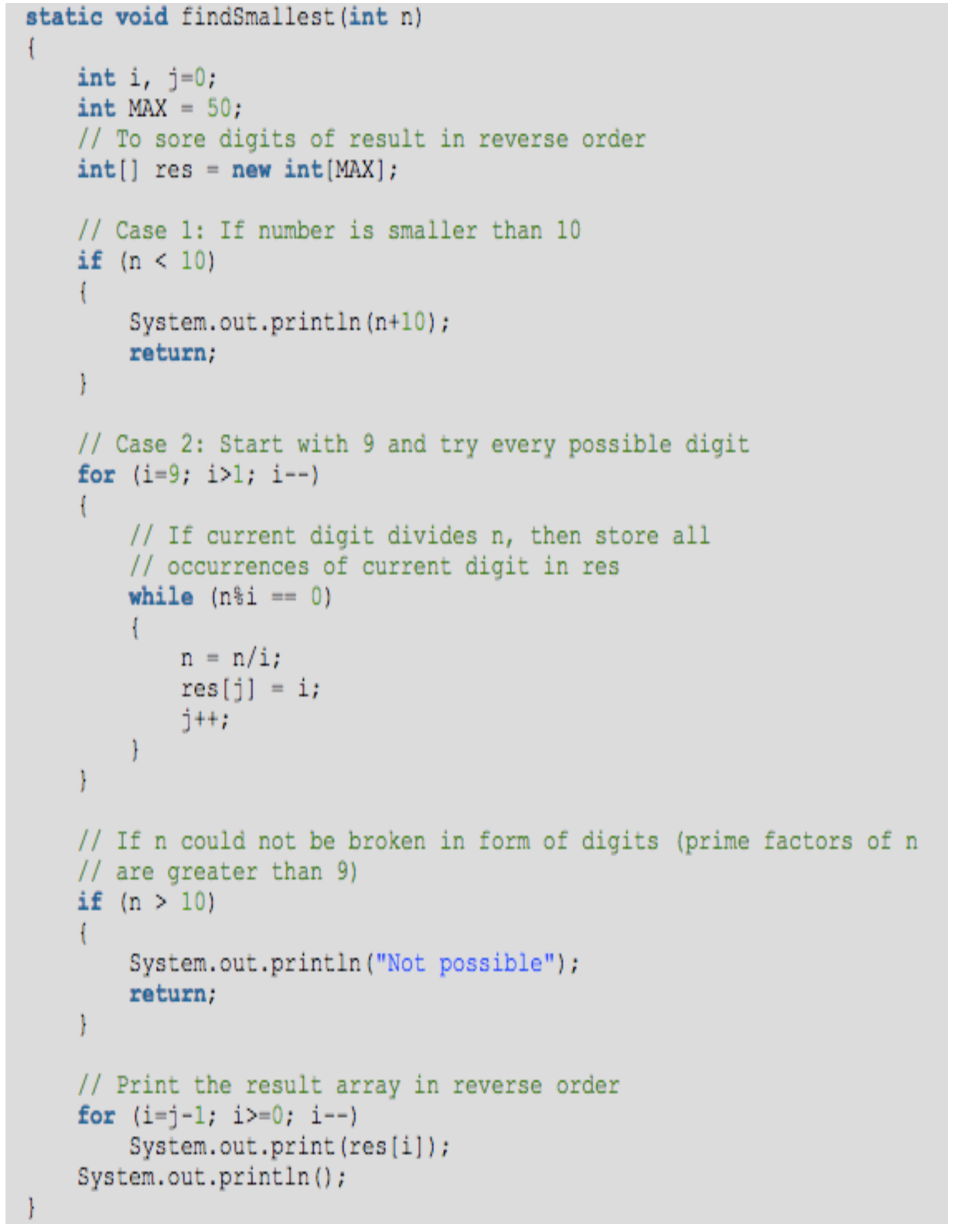
Sample exam – practical task

Do the following for the given source code:

a) Draw the graph (with clearly marked nodes)

b) Use the given graph to denote all def and use for the variable n

c) Find the minimum test set that achieves the All-du-paths-Coverage for n. Specify the procedure for creating all-du-paths in details.



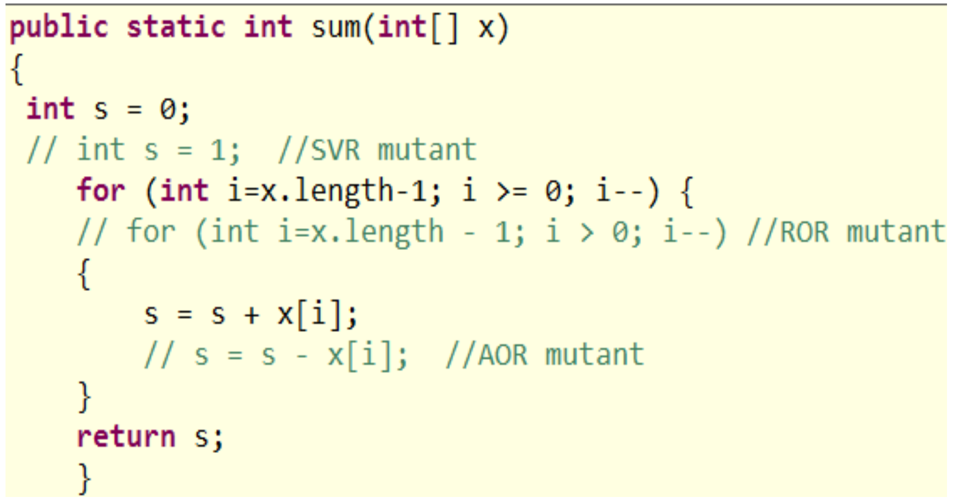
```
static void findSmallest(int n)
{
    int i, j=0;
    int MAX = 50;
    // To sore digits of result in reverse order
    int[] res = new int[MAX];

    // Case 1: If number is smaller than 10
    if (n < 10)
    {
        System.out.println(n+10);
        return;
    }

    // Case 2: Start with 9 and try every possible digit
    for (i=9; i>1; i--)
    {
        // If current digit divides n, then store all
        // occurrences of current digit in res
        while (n%i == 0)
        {
            n = n/i;
            res[j] = i;
            j++;
        }
    }

    // If n could not be broken in form of digits (prime factors of n
    // are greater than 9)
    if (n > 10)
    {
        System.out.println("Not possible");
        return;
    }

    // Print the result array in reverse order
    for (i=j-1; i>=0; i--)
        System.out.print(res[i]);
    System.out.println();
}
```

Sample exam – practical task

See the following code in which operational mutants are defined. The given code is a function that computes the sum of the elements of a string.

Answer the following questions:

- a) Find the conditions for reachability, infection and propagation for each mutant.
- b) Propose appropriate tests that kill mutants. Give a specific test with explanation.
- c) For each mutant individually, if possible, find values that meet infection but not propagation.
- d) Can you create an equivalent mutant for the given code? Please provide a test to confirm your answer.



```
public static int sum(int[] x)
{
 int s = 0;
 // int s = 1;  //SVR mutant
    for (int i=x.length-1; i >= 0; i--) {
    // for (int i=x.length - 1; i > 0; i--) //ROR mutant
    {
        s = s + x[i];
        // s = s - x[i];  //AOR mutant
    }
    return s;
    }
```

Sample homework - Selenium

Create a Selenium script that will create automated tests with the following requirements:

- Open the web site http://zero.webappsecurity.com/
- Log in accordingly using the following credentials: username, password
- Go to the Pay bills link
- Go to the Purchase foreign currency tab
- Select Canadian dollars
- Enter a value of 100
- Choose US dollars
- Click on Calculate Costs
- Confirm: $ 94.19 (CAD) = 100.00 U.S. dollar (USD)

Sample homework - JUnit

Write a function that checks whether a word is a palindrome

public Boolean palindrome (String word)

which returns a Boolean variable which if true indicates that the word is a palindrome and if false it is not a palindrome.

Using JUnit to do the following:

- Create tests for the given function;
- Answer whether the tests created reveal possible deficiencies and outliers of the written function;
- Fix the tests to find out the failures and errors of the written function.
- Create parameterized tests
- Create exception tests

Sample projects (chosen by students)

Unit testing an arbitrary job executor with Moq and Nunit; code coverage by using Coverlet..

Testing the e-Health system using Selenium and Mockito…

jStockMiner tool testing..

Youtube.com web page testing..

Cineplexx.mk web page testing

Sample seminar works (chosen by students)

- How to use Appium Tool
- TestComplete framework
- Robot framework
- Tool for automated testing Sikuli
- TestNG testing framework…

Evolution of student results

Year	Enrolled Students	Students with grade	Average Grade
2018/ 2019 *	88	8%	8.14
2017/2018	53	58%	8.19
2016/2017	38	71%	8.15
2015/2016	67	85%	8.14
2014/2015	38	79%	8.33
2013/2014	59	75%	8.02
2012/2013	37	62%	7.74
2011/2012	65	86%	7.34
2010/2011	67	/	/
2009/2010	64	/	/
2008/2009	68	/	/

Total number of enrolled students: 644

**only in the June session*

Invited lectures

- Endava – Skopje: UI Test automation using Selenium (JAVA)
- ITLabs – Skopje: Page Object Model Practical implementation with Selenium
- Endava – Skopje: API testing
- Seavus – Skopje: Java Technology Line Manager

Preparing students for software industry

- Direct communication with IT companies, learning from practice
- Invitations for summer internships with the possibility for employment
- Practical lessons and learning new testing tools
- Working on real projects

Acknowledgement

- Special thanks goes to my colleagues prof. Anastas Mishev and prof. Hristina Mihajloska for the successful local delivery of the course.
- We really appreciate the motivation of the companies to present their work and to offer internships and employments.

- Thank you for your attention.