ISSN: 2167-1435
v31 n3068

A Security Focused Undergraduate Program

Deborah Becker
dbecker@missouriwestern.edu

Ladan Kianmehr
lkianmehr@missouriwestern.edu

Wedad Almaghrabi, Graduate Student
walmaghrabi@missouriwestern.edu

Department of Computer Science, Mathematics and Physics
Missouri Western State University
Saint Joseph, MO 64507, United States

Abstract

Information security is a central topic in the news and academic circles. New models for software design are in production, but they have not achieved inclusion in computer science and computer information systems programs and textbooks. Microsoft's Security Development Lifecycle, McGraw's Security Touchpoints, and Wysopal's Secure Software Development Lifecycle all update the standard USE CASE to include misuse cases; focusing on the customer perspective of security requirements in application development. Typical undergraduate programs cover security topics in senior level network and software security courses. Applying security concepts to end products does not address core security issues. Building security concept objectives into entry level courses and propagating these concepts forward at every level of an undergraduate information technology curriculum will change the next generation of developers. ACM's, *Computer Science Curricula 2013,* includes a new line item, Information Assurance, which was not included in the 2001, and 2008, curricula guides. Revitalizing programs can be achieved through graduate research and updating opportunities in curriculum and textbooks. Grant money is being offered through NSF and the US Department of Education to bring security forward as an integral objective in existing and new programs. Information technology security is the motivation of new programs, models and regulations.

Keywords: security, software, information assurance

1. INTRODUCTION

Information security is a headliner in news publications, the central topic at conferences, the focus of new government regulations, and a part of everyday life. When educators and information technology professionals examine information security today, discussions include SQL injection, buffer overflow, cross-site scripting vulnerabilities, software language weaknesses, and database issues. Typical undergraduate programs cover these topics in senior level network and software security courses. Applying security concepts to end products does not address core security issues. Building security concept objectives into entry level courses and propagating these concepts forward at every level of an undergraduate information technology curriculum will change the next generation of developers.

ISSN: 2167-1435
v31 n3068

Prior to the internet revolution, security was primarily handled at the network level by hardware and operating system software. Today, security issues must be addressed at every point of attachment. New network configurations include wired and wireless hosts; network perimeters are not clearly defined and users connect globally. Security must start with software design, and be built into the network from the ground up.

2. REVIEW OF THE LITERATURE

The increase in employing information technology often accompanies an increase in its misuse or abuse, which necessitates the creation and development of procedures that protect information related machines and software. Incidents of identity theft, data theft, cyber-terrorism, and cyber-eavesdropping point to the demand for trained professionals who can meet the challenges of security in information technology. This predicts the need for highly trained information security professionals. However, the literature suggests that there exists a "lack of emphasis on security issues" in undergraduate curriculum (Bogolea & Wijekumar, 2004:59). According to these authors, until the turn of the millennium, software developing trainees (learners) lacked formal training on how to protect their products. Issues with system security are exacerbated by the Internet's contribution. Internet applications are susceptible to a wide range of attacks. Lack of security is universal; libraries, banking, online shopping etc. are all affected. For example, in a recent study of web security vulnerabilities using digital libraries, Kuzma (2010) found serious security flaws in the majority of the web applications employed by the libraries in the UK.

Wide-spread Internet security issues compel researchers to focus on vulnerabilities in an attempt to identify security issues afflicting software development. They have attempted to identify security requirements and strategies that evaded possible attacks. The sheer volume of research on this issue is a testament that software security is one of the most important issues facing application developers. Technologists strive to design software that can function smoothly, and evade harmful attacks. McGraw (2004), who has been a pioneer researcher on software security issues, contends that an important aspect of the computer security problem is software design. Design problems, poor problem specifications, faulty logic, and implementation errors comprise the system security issues. Internet security solutions, such as firewalls, have proven to be ineffective. McGraw and other groups have been developing new models for software and system design over the past decade.

New Design Models

Three new major models for software design and implementation are Microsoft's Security Development Lifecycle (SDL) (Microsoft, 2014); McGraw's Security Touchpoints (TP) (McGraw, 2005); and Wysopal's (IBM) Secure Software Development Lifecycle (SSDL) (Wysopal, 2006). These designs incorporate traditional USE CASE models, and structural design elements with additional focus on software security.

Microsoft (2014) has retooled their software development model to incorporate security from inception, see Figure 1. The SDL model uses a 'risk based approach *to guide software security investments through a program of continuous improvement…SDL became a company-wide mandatory policy in 2004'.* In 2011, this new approach to new software development became ISO/IEC 27034-1 compliant. The model incorporates a seven step iterative approach to projects: train personnel, design requirement benchmarks (quality gates), create application design, approve design tools, verify design with attack reviews, release, and respond.

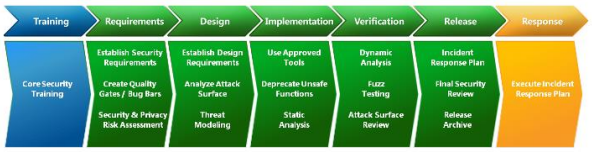


Figure 1, Secure software development process model at Microsoft (Microsoft, 2006)

The two final steps include software updates that are pushed down to installations through internet connections. These patches are in response to user reported problems with products and internet attack episodes.

Dr. Gary McGraw (2005), has contributed a new model, Security Touchpoints, a six-phase model applied to the software development lifecycle.

ISSN: 2167-1435
v31 n3068

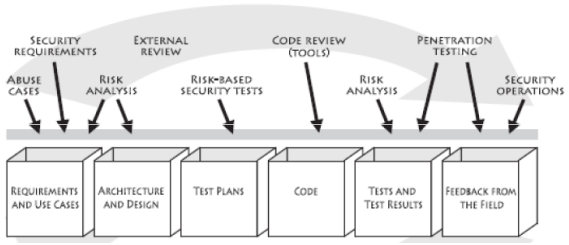


Figure 2, Security Touchpoints

This new model extends the usual USE CASE model incorporating a misuse case level, see Figure 2. The premise behind the model is that 50% of security defects in software come from implementation bugs, the remaining 50% were actually built into the application through design flaws. Bugs include such items as buffer overflow, unsafe environment variables, race conditions, unsafe system calls, Cross-site scripting, SQL injection; design flaws include: misuse of cryptography, compartmentalization problems in design, catastrophic security failure (fragility), type safety confusion error, insecure auditing, broken or illogical access, and method over-riding problems(subclass issues).

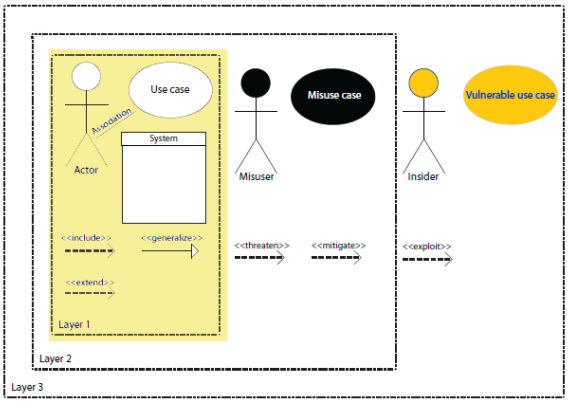


Figure 3, A new USE CASE Model

In Figure 3, we have drawn McGraw's USE CASE diagram. The additional 3rd level incorporates insider and attacker scenarios.

Wysopal's (2006) Secure Software Development Lifecycle model, developed through years of service at IBM, takes a mainframe approach to software security. This model includes infrastructure hardware services as an integral part of the development system. Mainframe applications rely heavily on system security integration; legacy COBOL programs require access control to be handled at the firewall gate.

Figure 4, Secure Software Development Lifecycle

The five phases of development are: requirements specification; architectural design, review and threat modeling; coding, which includes best practices and static analysis; testing for vunerability assessment; and deployment that includes server configuration and network configuration reviews. Figure 5, shows a more modern view of the phases.

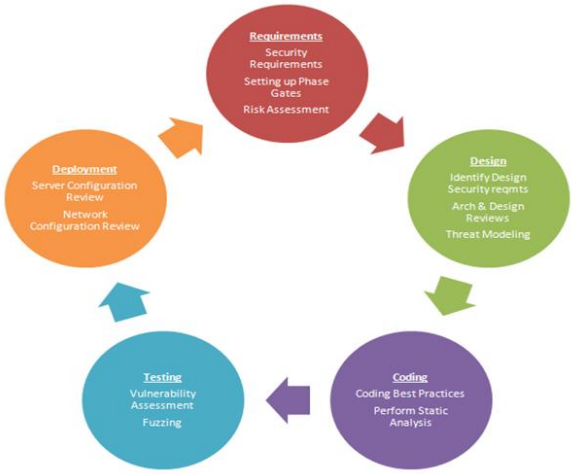


Figure 5, Wysopal, SSDL Five Phases

The concepts of these models are not new but retooled to include security at every phase of development. Inclusion at entry levels will ensure deeper understanding of security's role in application development.

While, it is not an earth shattering discovery that security is needed and must be guaranteed by software applications that provide multi-faceted functionalities; our ability to transfer data from one host to another in performing these functionalities has made software more susceptible to penetration in terms of confidentiality, accountability, and integrity (Jung, Kim, Masoumzadeh, & Joshi, 2012). Software developers must learn how to integrate the security handshake between operating systems and applications programs (Joshi, 1987). They collectively direct the operation of

a computer and execute the commands received from external sources. Software security researchers (e.g., Jung et al., 2012; Haider, Magnusson, Yngstrom, & Hemani, 2011; Gordon, Loeb, & Lycyshin, 2003) recommend that successful software developers must pay attention to the "hows" of managing external factors. Rapid technological improvements make the security of the system in place obsolete and vulnerable. Constant updating is costly; building security into applications from inception can mitigate some of these costs.

The literature suggests educators in information systems development programs; enrich programs of study by focusing on software security issues from a systematic point of view. Heckman, (2003), believes this will reduce security flaws in software development at its origin. Integrating security principles into software takes a consumer perspective; enhancing the integrity of the products. Program requirements specification guidelines, and best practice requirements incorporate control and consistency in application development (Abu Talib, Khelifi, and Jololian, 2010). Educators must guide students in the process insuring close attention to the design of software. Security must be instilled in them from the very start of their career. Design and programming courses must incorporate system and software testing techniques (Jarzombek & Goertzel, 2006). The main emphasis here is avoiding vulnerabilities by practicing better coding techniques.

Although the common thread in the literature reviewed here is an emphasis on some sort of training program or curriculum on security for software developers; the type of training, and its content, has been discussed in the literature only sporadically. This necessitates a form of formalized university level education for professionals of information systems and software developers (Sheoran, Friesen, & de Belón, 2006). Thus, articulating a training program early on at the undergraduate level is essential in maintaining the integrity of the discipline. However, since there is no consensus regarding the body of knowledge that software developers must have while learning the trade at its rudimentary level, this paper proposes the integration of teaching security programming into a beginning level undergraduate courses.

New Curriculum Guidelines

ACM released, 'Computer Science Curricula 2013' on December 20, 2013. Security figures prominently in the guidelines for existing and new CS/CIS/MIS curriculum. They recommend one hour of a three hour (1/3) Introduction to Programming course covering secure coding practices and concepts. They infer that first and second year courses should deal with security in both design and development. See an example assignment in the attachment; the assignment simply adds error checking controls to make the resulting program more secure. ACM also suggests that security concepts and practices consume another six hours in junior and senior level courses. Information assurance and security was clearly missing from the 2001 and 2008 curricula guides (ACM, 2013). Since 2008, there has been a paradigm shift to internet based software-as-service applications. The access points in existing curriculum are self-defined. Updating existing lecture materials and lab assignments to include security objectives will enrich undergraduate programs.

3. INTEGRATION – MOVING FORWARD

Information technology disciplines, the learning "by" disciplines, are teaching the learning "by" generations. Generation X, the Millennials, and Generation Z, learn by hearing, example, sharing, seeing, mimicking and they embrace technology innovations. They share and collaborate on assignments and assessments with or without permission. The new conceptual model of the 'flipped classroom' understands this. Assigning activities and quizzes as before class events leaves class time for learning by example and collaboration. Student working together to solve problems that incorporate the new security objectives will propagate these new objectives across IT undergraduate programs.

Evaluating existing programs is the first step to security integration. Touch points include entry-level programming and design courses. Most programs have junior and senior level courses addressing security at the network level. These courses serve a two-fold purpose: introduction of security concepts in both network and software topics, and examples of applications. When security is built into entry-level courses and propagated throughout the degree, discussions and labs in upper level courses are broader, concepts are better understood, students exhibit deeper connections in assignments and assessment tools, and perform

ISSN: 2167-1435
v31 n3068

better in entry-level development jobs (McGraw, 2006).

Revitalized Courses

The typical undergraduate computer information systems (CIS), computer science (CS), and management of information systems (MIS) programs include courses in computer concepts, high level languages, data structures, database, systems analysis, operating systems, and other courses filling out a well-rounded curriculum. Junior and senior years are peppered with object oriented and computer security courses to enhance programs.

Junior colleges and technical institutions offer focused computer security degrees funded by grant money offered by the National Science Foundation (NSF) and US Department of Education. CIS/CS/MIS degrees are popular workforce reentry programs. Market need for technical savvy computer security professional make these two year computer security technical degrees attractive.

Security should not be a technical degree or an afterthought. Incorporating security concepts into existing courses through inclusion of security leaning objectives broadens the scope and deepens understanding of security's role in development. Security concepts can be easily worked into existing assignments and assessment tools. Graduate research can be focused to aid in the revitalization of existing course materials. New teaching assistants gain perspectives, making them valuable academic candidates for information technology programs.

Other opportunities in the security integration forum include incorporation of these objectives into course textbooks. Although, these new models have been in the literature for nearly a decade, they are not prevalent in current textbook offerings (McGraw, 2006). New case studies and end of chapter problems integrating these new models will add context. Students are open reservoirs, accepting direction from progressive faculty. The inclusion of these additional security components will seem logical. Like recycling, and global warming, these models will form the new standards of accepted development. Today's student is more technically oriented and the concept of a global network perimeter is natural; security already occupies a prominent place in their life activities.

Design Tools Address Security

Our language tools are changing. Java, JScript, PHP5, Ajax, Ruby, C#, Objective-C, C++, HTML5, CSS3, and others are included in information technology programs; web development now holds a prominent place in most modern curricula (http://mashable.com). New, more advanced APIs like Google tools, and Twitter's bootstrap integrate programs, and they are attractive to the generations of students entering college. These new APIs have addressed security; issuing written security guides for implementation. New curriculums are including web development as an integral part of undergraduate programs. Java development for the web includes these new APIs.

Undergraduate CIS programs embraced Java as an introductory language in early 2000s. Programs moved away from C and C++, because of the complexity, and the feelings of some educators that C++ was too near the core operating systems; giving young developer students access to vulnerable areas in information systems. Java is easy to learn and a modern object oriented language (Blake, 2011). The switch to Java, however, has had its drawbacks.

In the past eighteen months; Oracle, Java's parent, has released patches covering over 180 security problems identified by users and hackers (Bell, 2014). Like IBM, Oracle is a mainframe based organization. Entrance into the open source community came with the purchase of tools such as MySQL and Sun Microsystem's Java. These tools bring problems prevalent in open-source applications. There were discussions in 2013, about moving away from Java in academia, however, this tool has been integrated into Adobe and, Microsoft applications and drives the major web presence on the internet. Students entering the new job market are expected to understand Java, Java Script, and the new API models.

4. EXAMPLE JAVA I ASSIGNMENT

See attachment: Example Assignments

5. SUMMARY

It's an exciting time in information technology; new models, new APIs, new languages. Updating opens new opportunities for students and faculty. The integration of security concepts

into all courses in information technology programs will deepen and broaden the understanding of security's role in the development cycle. Undergraduates can help in the updating of existing work through independent study projects. Graduate Assistants can be involved in the research; how and where to implement the new security guidelines. This is just a make-over; all programs need a fresh face. We are training the next generation of innovators who are facing challenges unimagined in computer science a mere decade ago.

6. REFERENCES

Abran, A., Moore, J.W., Bourque, P., Dupuis, R., and Tripp, L. (2004). *Software Engineering Body of Knowledge.* Los Alamitos: IEEE Computer Society Press. Extracted from Abu Talib, M., Khalifi, A., Jololian, L. (2010). Secure Software Engineering: A New Teaching Perspective Based on the SWEBOK. Interdisciplinary Journal of Information, Knowledge, and Management, 5: 83-99.

Abu Talib, M., Khalifi, A., Jololian, L. (2010). Secure Software Engineering: A New Teaching

Bogolea, B., & Wijekumar, K. (2004). Information security curriculum creation: A case study. InfoSecCD Conference presentation, October, Kennesaw, GA, USA.

Gordon, L., Loeb, L., & Lycyshin, L. (2003), Information Security and Real Options: A Wait-And-See Approach. *Computer Security Journal*, 19(3): 1-8.

Haider, A., Magnusson, C., Yngstrom, L., & Hemani, A. (2011) Addressing Dynamic Issues in Information Security Management. *Information Management & Computer Security*, 19(1): 5-24.

Heckman, C. (2003). Two Views on Security Software Liability: Using the Right Legal Tools. *IEEE Security & Privacy Magazine*, 10(1): 73-75.

Joshi, B. (1987). *U.S. Patent No. 4,688,169*. Washington, DC: U.S. Patent and Trademark Office.

Jung, Y., Kim, M., Masoumzadeh, A., & Joshi, J. (2012). A Survey of Security Issue in Multi-Agent Systems. *Springer Science+Business Media,* 37: 239-260.

Kuzma, J., (2001). European Digital Libraries: Web Security Vulnerabilities. *Library Hi Tech*, 28(3): 402 – 413.

Linder, F. (2006). Software Security is Software Reliability. Communication of the ACM – Hacking and Innovation (magazine), 49(6): 57-61.

McGraw, G. (2004). Software Security. Security & Privacy, IEEE, 2(2), 80-83.

McGraw, G. (2006). Software Security: Building Security In (Vol. 1). Addison-Wesley Professional.

Sheoran, P., Friesen, C., & de Belón, H. (2006). Developing and Sustaining Information Assurance: The Role of Community Colleges, part 2. IEEE Security and Privacy, 4, 60–65.

Microsoft, (2014), "The Security Development Lifecycle." *Microsoft MSDN Blog*. Microsoft, 14 May 2003. Web. 17 Jan. 2014. <http://msdn.microsoft.com/en-us/library/cc307406.aspx>.

McGraw, Gary. (2005). "The 7 Touchpoints of Secure Software." Dr. Dobb's, The World of Software Development. Dr. Dobbs, 1 Sept. 2005. Web. 03 Feb. 2014.

Wysopal, C. (2006)."Introduction to Secure Software Development Life Cycle - InfoSec Institute." *InfoSec Institute*. InfoSec Institute, 01 Jan. 2012. Web. 03 Feb. 2014. <http://resources.infosecinstitute.com/intro-secure-software-development-life-cycle/>.

Introduction to Secure Software Development Life Cycle - InfoSec Institute. (n.d.). Retrieved from http://resources.infosecinstitute.com/intro-secure-software-development-life-cycle/

ACM, (2013). The Joint Task Force on Computing Curricula Association for Computing Machinery (ACM) IEEE Computer Society. (2013, December). Computer Science Curricula 2013. http://http://www.acm.org/education/CS2013-final-report.pdf

ISSN: 2167-1435
v31 n3068

Bell, L. (2014). Oracle to issue huge security patch addressing 36 Java vulnerabilities- The Inquirer. http://ww.theinquirer.net/inquirer/news/2322481/oracle-to-issue-huge-security-patch-addressing-36-java-vulnerabilities

Blake, J. (2011). Language considerations in the first year CS curriculum. Journal of Computing Sciences in Colleges. (Vol 26). 124-129.

ISSN: 2167-1435
v31 n3068

7. Attachment

Assignment: File IO

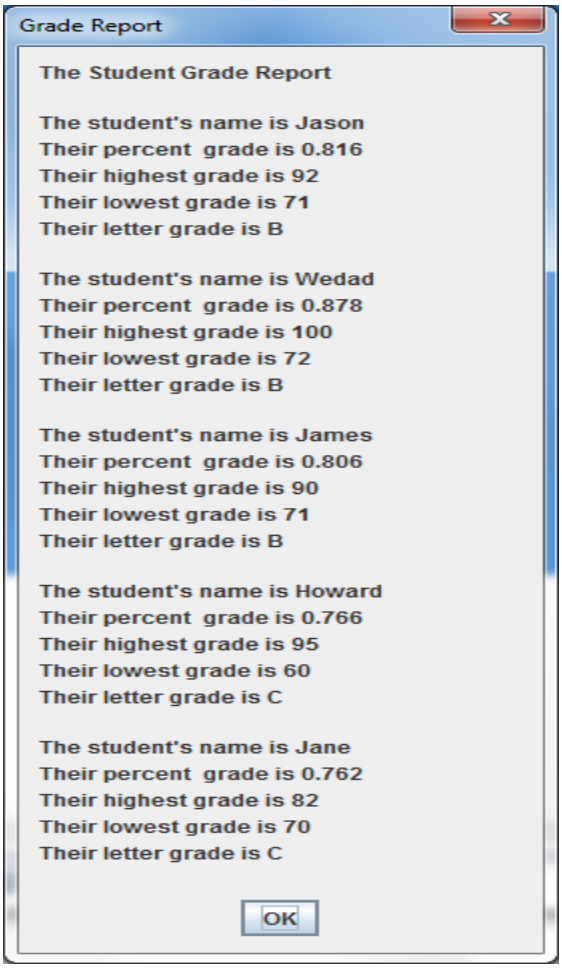
Write a java program that reads a text file called 'grades.txt' containing student grades. The data file is located the class Moodle folder under Assignment 4 and contains a series of records. Each record contains a student name followed by five grades. Using looping structures read all the records from the file. Do the following: sums the student's total score, calculates the student's average grade, finds their highest grade, their lowest grade and assigns the student a letter grade.

Create a disk report (report.txt) that displays all the information for each student. Also create a grade report that displays the report to the screen using the JOptionPane class. Sample reports might look like the following:

Disk Report

```
THE STUDENT GRADE REPORT

The student's name is Jason
Their percent grade is 0.816
Their highest grade is 92
Their lowest grade is 71
Their letter grade is B

The student's name is Wedad
Their percent grade is 0.878
Their highest grade is 100
Their lowest grade is 72
Their letter grade is B

The student's name is James
Their percent grade is 0.806
Their highest grade is 90
Their lowest grade is 71
Their letter grade is B

The student's name is Howard
Their percent grade is 0.766
Their highest grade is 95
Their lowest grade is 60
Their letter grade is C

The student's name is Jane
Their percent grade is 0.762
Their highest grade is 82
Their lowest grade is 70
Their letter grade is C
```

JOptionPane Report



Your program must follow these guidelines:

1. Comment your program telling what each object and structure is doing.
2. Use the correct data types for the output needed.
3. Correct all runtime and syntax errors.
4. Format your output to display the data in a professional manner.
5. For a grade submit, the java code, the data file grades.txt, the executed report.txt, and a word document showing the screen shot of your JOptionPane report.

ISSN: 2167-1435
v31 n3068

Assignment: File IO – With Security

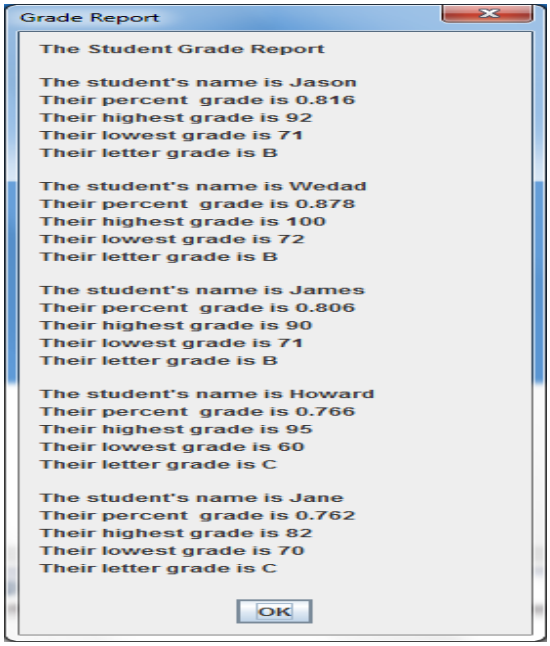
Write a java program that reads a text file called 'grades.txt' containing student grades. The data file is located the class Moodle folder under Assignment 4 and contains a series of records. Each record contains a student name followed by five grades. Using looping structures read all the records from the file. Do the following: sums the student's total score, calculates their average grade, find their highest grade, their lowest grade and assigns the student a letter grade.

Create a disk report (report.txt) that displays all the information for each student. Also create a grade report that displays the report to the screen using the JOptionPane class. Sample reports might look like the following:

Disk Report

```
THE STUDENT GRADE REPORT

The student's name is Jason
Their percent grade is 0.816
Their highest grade is 92
Their lowest grade is 71
Their letter grade is B

The student's name is Wedad
Their percent grade is 0.878
Their highest grade is 100
Their lowest grade is 72
Their letter grade is B

The student's name is James
Their percent grade is 0.806
Their highest grade is 90
Their lowest grade is 71
Their letter grade is B

The student's name is Howard
Their percent grade is 0.766
Their highest grade is 95
Their lowest grade is 60
Their letter grade is C

The student's name is Jane
Their percent grade is 0.762
Their highest grade is 82
Their lowest grade is 70
Their letter grade is C
```

JOptionPane Report



Your program must follow these guidelines:

1. Comment your program telling what each object and structure is doing.
2. Use the correct data types for input and output as needed.
3. Correct all runtime and syntax errors.
4. Format your output to display the data in a professional manner.
5. Add a try-catch block that displays user defined errors for the following conditions.

ISSN: 2167-1435
v31 n3068

   a. Display a NoSuchElementException elementException error if the program reads beyond the end of file or record.
   b. Display a FileNotFoundException error if the programs cannot find the data file.
6. Test your error messages by doing the following:
   a. Rename your input file grades2.txt in the program.
   b. Instruct the loop to read data beyond the record limit. EXAMPLE: Use a **for-next loop**, design your loop header to read 6 grades – for (int i=0; i<=5; i++){
   c. Record the error messages with screen shots, paste them into a word document.
7. For a grade submit, the java code, the data file **grades.txt**, the executed **report.txt**, and a word document showing the screen shots of your JOptionPane report and error messages.