**TEXAS STATE UNIVERSITY COURSE ADDITION FORM**

1. **Effective Semester:** Fall 2016

2. **College:** Science and Engineering

3. **Department/School/Program:** Computer Science

4. **Prefix/Subject Number**
   - CS 3354

5. **Course Title:**
   - Proposed Long: Object-Oriented Design and Programming
   - Abbreviated (18 characters only including spaces): OBJ - ORTD DSG & PRG

6. **Course Description (complete sentences in 50 words or less):**
   The course covers object-oriented design principles and programming for students with prior programming experience. The topics include inheritance and polymorphism, object-oriented design process, UML diagrams, design patterns, exception handling and multithreading. Students will design and implement programs in Java. Cannot receive credit for this course if student took CS 4354. Prerequisite: CS 2308 with a grade of C or higher or consent of instructor.

7. **Prerequisites (Including Minimum Grade Required):**
   - CS 2308 with a grade of C or higher or consent of instructor.

8. **Co-Requisites (Including Concurrent Enrollment Allowed):**

9. **Restrictions:**
10. Course Data

**CIP Code (10 digits - no spaces or periods)**

```
1 1 0 7 0 1 0 0 0 6
```

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Lecture Contact Hours</th>
<th>Lab Contact Hours</th>
<th>Credit Hours</th>
<th>Repeatable for Credit?</th>
<th>Maximum Credit Hours Allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-Lecture</td>
<td>x 3</td>
<td>0</td>
<td>3</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>2-Lab</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-Practicum/Internship</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-Seminar</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-Independent</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6-Private Lesson</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-Thesis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9-Dissertation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0-Individualized</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-Clinical</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Writing Intensive?</th>
<th>Topics Course?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>x No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Valid Grade Mode (choose only one)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Letter</td>
</tr>
<tr>
<td>Credit/ No Credit</td>
</tr>
<tr>
<td>Leveling/Assistantships/ESL</td>
</tr>
<tr>
<td>Developmental</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Course Equivalency(s) (Prefix and Number)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS 4354</td>
</tr>
</tbody>
</table>

11. Justification for the course action:

- **Degree:** Computer Science
- **Major:** Computer Science
- **Minor:**
- **Certificate:**

This new course replaces CS 4354. Modern software engineering requires students to understand the fundamentals of object-oriented programming (OOP). At the junior level, this required new course allows students to learn these concepts earlier in the curriculum. It will also better prepare, or earlier qualify, the students for internships and undergraduate research opportunities which require the skills learned in the course. Offering the course content at a level lower than 4000 was recommended during the fall 2013 ABET accreditation visit for the BS in CS.
12. Course Goals and Objectives:
- Must be specific and unique to each course.
- Must be stated in measurable terms.
- Must have distinct differences between a graduate level course and an undergraduate course (in case of stacked courses).
- Please refer to Bloom's Taxonomy of Measurable Verbs.

At the end of the semester the student should be able to:
1. Design, implement, test, and debug programs written in Java.
2. Describe inheritance and polymorphism and use them in Java programs.
3. Describe exception handling in Java, and use it to write reliable Java code.
4. Read and write Java programs that use threads to implement concurrency.
5. Read, design, and draw the following models using the Unified Modeling Language (UML): Class diagrams, Sequence diagrams, State diagrams.
6. Write Java code that implements the designs specified by UML diagrams.
7. Describe (at least) the following Design Patterns: Adapter, Strategy, Command, Composite, Observer; create UML designs using them, and implement the designs in Java programs.
8. Determine the proper design pattern for a given problem specification.

13. Description of Instructional Methodologies:
- Examples include lecture, discussions, group projects, role playing, simulations, modeling, field-based activities, writing, cooperative learning, inquiry, experimentation, product design, creative activities, case studies, seminars, internship activities, coaching, etc.

Lecture, discussions, individual and/or group projects, product design, modeling

14. Assessment of Student Learning:
- Examples include tests, projects, presentations, performances, creative works, papers, etc.
- Above examples of assessment must include percentages of total grade assigned.
- Must have distinct differences between a graduate level course and an undergraduate course (in case of stacked courses).

Written exercises (models, design documentation) 15%
Programming assignments (Java programs) 20%
Exams 30%
Final Exam 35%
15. Course Outline:
- Provide a weekly outline as appropriate for an example semester in which the course will be taught
- Must distinguish the course clearly from similar offerings in the same or other programs.
- Must indicate specific topics.

<table>
<thead>
<tr>
<th>Week 1: Java Introduction</th>
<th>Week 9-10: Design Patterns: Adapter, Strategy, Command, Composite, Observer and MVC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 2: Java I/O and Serialization</td>
<td>Week 11-14: User Interface programming: Swing framework as an example of inheritance and design patterns</td>
</tr>
<tr>
<td>Week 3: Java Exceptions and Collections</td>
<td>Week 15: Review</td>
</tr>
<tr>
<td>Week 4: Java Inheritance and Polymorphism</td>
<td>Week 16 (Finals Week): Final</td>
</tr>
<tr>
<td>Week 5-6: Object Oriented Design Process and UML Diagrams: Class Diagrams, Sequence Diagrams, State Diagrams</td>
<td></td>
</tr>
<tr>
<td>Week 7-8: Java Threads</td>
<td></td>
</tr>
</tbody>
</table>

16. Suggested Textbook(s) and Other Learning Resources:
- Must list the required and recommended (if any) resources (e.g., relevant textbooks, course packets, websites), with complete bibliographical data (author, title, date and other publication data) in a standard academic format (e.g., CBE, APA, MLA, Chicago, etc.)

Type: Required

17. Bibliography:
• Must include literature other than required textbooks and other learning resources.
• Must demonstrate familiarity with current research. Ordinarily, the bibliography should include scholarship published during the last five years.
• Must conform to a standard academic format (e.g., CBE, APA, MLA, Chicago, etc.) Each bibliography will use only one format.

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1995. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.


Walter Savitch. 2012. Absolute Java (5th ed.). Addison-Wesley Publishing Company, Boston, MA, USA.


18. Approvals:

Department Chair/Program Director/School Director

[Signature]

Date: 1/28/2015

Chair of College Curriculum Committee

[Signature]

Date: 3/3/2015

Dean of College

[Signature]

Date: 3/4/15

Dean of The Graduate College (if applicable)

Date

Chair of University Curriculum Committee (if applicable)

Date