CS 121 Midterm Exam
Fall 12

CLOSED BOOK
THE RULES - PLEASE READ CAREFULLY

- Due Saturday, 13 Oct, 9 PM.
- Take this test in a SINGLE 1.5 hour setting.
- This test is: CLOSED Book - NO textbook, NO notes, etc.
- If you do not understand a question, then write down your understanding along with your answer.
- You CANNOT use a computer.
- You CANNOT use course tests, notes, etc., from any previous version of the course.
- Also, **please refrain from discussing any aspect of this exam among students until after the tests are returned**
- Give yourself up to 10 minutes to just read over the test - don't write anything during this period. Immediately afterwards, give yourself up to **1.5 uninterrupted hour** to take the test.
- As soon as you are finished with the test, return it to my bin outside the CS Dept Office or my desk in Sprague.
- Please fill in the information below. Make sure to sign your exam to indicate that you have taken the exam in accordance with the HMC Honor Code.
  - NAME: ____________________
  - Date: ____________________
  - Started Taking Test: ____________________
  - Finished Taking Test: ____________________
  - Signature: ____________________

- **1.** How do Iterative and Agile software development models differ from the Waterfall model?
  Waterfall:
  A classically linear and sequential approach to software design and systems development, each waterfall stage is assigned to a separate team to ensure greater project and deadline control, important for on-time project delivery. A linear approach means a stage by stage approach for product building. Waterfall requires speculative reasoning/decisions which do not get revisited until far later in the product development cycle.

  Agile:
  Four principles that constitute Agile methods are:
  1. The reigning supreme of individuals and interactions over processes and tools.
  2. As does, working software over comprehensive documentation.
  3. Likewise, customer collaboration over contract negotiation.
  4. And again, responding to change over plan follow-throughs.

  Differences
  To synopsize the difference between the two, one can say the classic waterfall method stands for predictability, while Agile methodology spells adaptability. Agile methods are good at reducing overheads, such as, rationale, justification, documentation and meetings, keeping them as low as is possible. Agile methods benefit small teams with constantly changing requirements, rather more than larger projects. Agile, based on empirical rather than defined methods (i.e., Waterfall) is all about light maneuverability and sufficiency. Agile methodology means cutting down the big picture into puzzle size bits, fitting them together when the time is right e.g. design, coding and testing bits.

  Since Agile provides flexibility to make changes as per customer requirements it is more inclined towards better client satisfaction. This is a real set back for the Waterfall model which does not allow any modifications once the module has been completed. Under Agile development modular partitioning of the software can be effectively carried. Agile demands repetition/iteration of the various stages while in the Waterfall each stage happens only once.

- **2.** Why are requirements so difficult to specify?

  Users do not understand what they want or users don't have a clear idea of their requirements
  Requirements change
  Users will not commit to a set of written requirements
  Users insist on new requirements after the cost and schedule have been fixed
  Communication with users is slow
  Users often do not participate in reviews or are incapable of doing so
  Users are technically unsophisticated

Users do not understand the software development process
Users do not know about present technology

- **3.** Describe the types of requirements denoted by FURPS+. Give an example of each in your Project. If there is no example, explain why?

  FURPS:
  Functional - feature set, security
  Usability - human factors, aesthetics
  Reliability - accuracy, mean time to failure
  Performance - speed, efficiency, throughput,
  Supportability - Testability, adaptability, portability

- **4.** How are requirements created and managed in Agile based software development?

  Requirements as envisioned in Waterfall, do not exist in Agile. That is: The word requirement carries the connotation of required, absolute, permanence. Agile views the most effective method of conveying information as face-to-face. Agile begins with 'User Stories' from which flexible requirements are created in the form of Use Cases. Some are fixed, e.g., Performance, but most are flexible and re-evaluated. The Goal stack is used to management requirements.

- **5.** What are key components of a Use Case? What is the purpose of each?

  Full blown

  Title:
  Primary Actor
  Goal in Context
  Initiator
  Scenario
  Scope
  Level

Stakeholders and Interests
Precondition
Minimal Guarantees
Success Guarantees
Trigger
Main Success Scenario
Extensions
Technology & Data Variations List
Related Information

- **6.** What is the difference between a Domain Model and a Class Model?

  The domain model identifies the relationships among all the entities within the scope of the problem domain (both behavior and data) and commonly identifies their attributes. The domain model provides a structural view of the domain that can be complemented by other dynamic views, such as use case models. An important advantage of a domain model is that it describes and constrains the scope of the problem domain. The domain model can be effectively used to verify and validate the understanding of the problem domain among various stakeholders.

  Class diagrams/models are the mainstay of object-oriented analysis and design. Class diagrams show the classes of the system, their interrelationships (including inheritance, aggregation, and association), and the operations (methods(and attributes of the classes. Class diagrams are used for a wide variety of purposes, including both conceptual modeling and detailed design modeling. Class diagrams are the next level of detail below Domain Models.

- **7.** We are building a miniature golf game. When the ball moves I need to detect collisions. Should collision detection be a responsibility of the ball class or its own class or some other class? Justify your answer in terms of good design principles and patterns.

  Collision detection should be its own class in order to obey the single responsibility principle. The attributes of a ball include, size, color, etc. However a collision is not really an attribute of a ball so it should not be part of the ball class. In addition we might want to include collisions between other entities, and thus collisions within the ball class would violate the principle of DRY.

- **8.** The *Open Close Principle* is a design principle that be applied to classes, modules, and functions. What does it mean?

  The open/closed principle states ”software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification”;
  that is, such an entity can allow its behavior to be modified without altering its source code.

  The idea was that once completed, the implementation of a class could only be modified to correct
  errors; new or changed features would require that a different class be created. That class could reuse coding from the original class through inheritance. The derived subclass might or might not have the same interface as the original class.

  9. *GOMS* is used as a model for analysis of UIs. What are the four facets of *GOMS*.

  Goals:
  are what the user intends to accomplish.
  Operators:
  are actions that are performed to reach the goal.
  Methods:
  are sequences of operators that accomplish a goal. There can be more than one method available to accomplish a single goal;
  Selection rules:
  are used to describe when a user would select a certain method over the others.

- **10.** Write a question and answer for any of the Software Development material we have covered.