

Problem 1 Software Development & Testing (20 pts)

- a. (4 pts) What is the main reason many software projects fail?
 - a. Poorly trained programmers
 - b. Insufficient project funding
 - c. Complexity of projects**
 - d. Slow computers
 - e. Insufficient computer memory

- b. (4 pts) What is the software life cycle?
A sequence of essential operations necessary for producing quality software.

- c. (4 pts) What is the first phase of the software life cycle?
 - a. Testing
 - b. Coding
 - c. Design
 - d. Specification**
 - e. Documentation

- d. (4 pts) According to the waterfall model...
 - a. Design all algorithms before coding **T or F**
 - b. Write test cases before coding **T or F**
 - c. Use prototype implementation to refine design **T or F**

- e. (4 pts) According to the unified model...
 - a. Design all algorithms before coding **T or F**
 - b. Write test cases before coding **T or F**
 - c. Use prototype implementation to refine design **T or F**

- f. (4 pts) Compared to program verification, empirical testing...
 - a. Handles larger programs **T or F**
 - b. Always catches more errors **T or F**
 - c. Ensures code is correct **T or F**
 - d. Can be applied without examining code **T or F**

- g. (4 pts) Program testing
 - a. Black box testing requires good programmers **T or F**
 - b. Code coverage is a measure of code testing **T or F**
 - c. Pre-conditions and post-conditions are used for empirical testing **T or F**

Problem 2 Object-Oriented Design (20 pts)

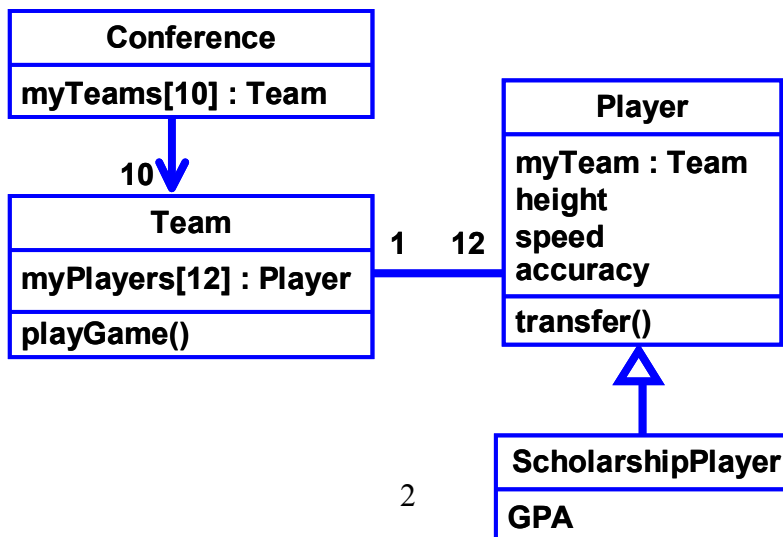
- h. (4 pts) State and behavior are two main qualities of objects in an object-oriented system.
 - a. What is the third quality? **Identity**
 - b. What is it used for? **Mechanism to distinguish between objects**
 - c. What is an example of its use in Java? **Reference variables, ==, .equals(), “this”**

- i. (4 pts) Object oriented design...
 - a. Produces faster programs T or F
 - b. Produces smaller programs T or F
 - c. Produces software without errors T or F

- j. (4 pts) Abstraction and encapsulation are two principles of object-oriented design
 - a. Define abstraction
Providing a simple high-level view of an entity or activity.
 - b. Define encapsulation
Confining / hiding information so it can only be accessed through a defined interface.
 - c. Describe how object-oriented design supports encapsulation
Data is hidden inside objects and can only be accessed through its public methods.

- k. (4 pts) Given the following problem description, produce an object-oriented solution. Include as many details as possible. Draw a UML class diagram (you may write code for Java classes ONLY if you don't know UML).

Design a simulation of a basketball conference. Each conference has 10 teams. Each team has 12 players. Each player has a specific height, speed, and accuracy. Players know which team they belong to. Some players are scholarship players. Scholarship players need to record their current grade-point average. Players may be transferred between teams. Teams play basketball games against other teams in the conference. The result of each game is determined using a function based on the height, strength, speed, and accuracy of the players on each team.

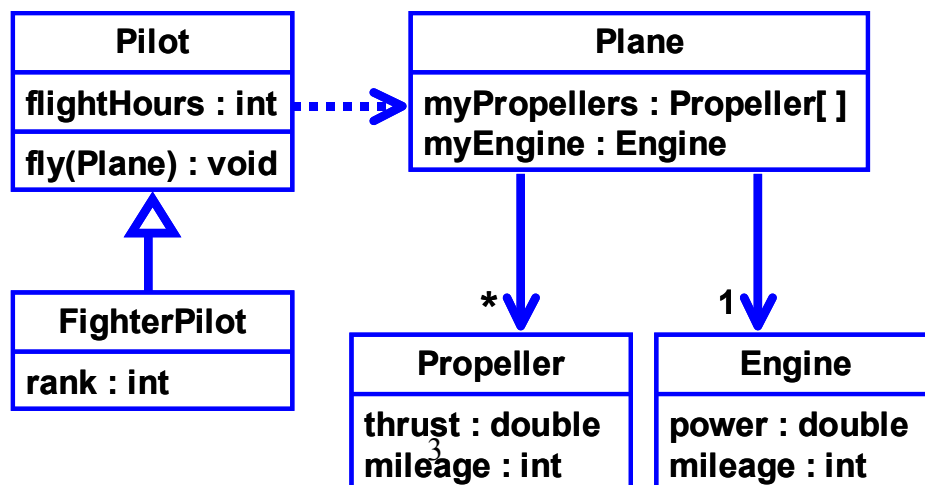


Problem 3 Unified Modeling Language (20 pts)

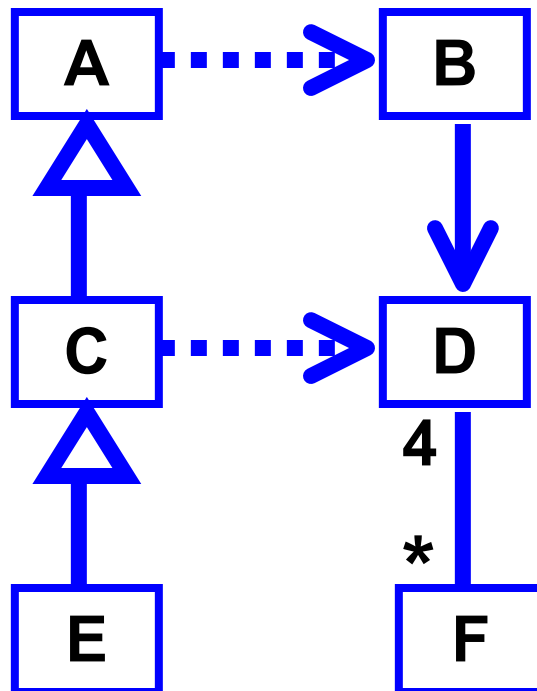
- l. (4 pts) Consider UML
 - a. What are class diagrams used for?
Describes the static structure of classes in a software system
 - b. What is an association?
A permanent, structural relationship between two classes A & B created when the state of class A contains objects of class B
 - c. What is a dependency?
A temporary relationship between two classes A & B created when a method in class A uses objects of class B

- m. (4 pts) Given the following Java code, draw a UML class diagram (you may write code for Java classes ONLY if you don't know UML).

```
public class Propeller {  
    public double thrust;  
    public int mileage;  
}  
public class Engine {  
    public double power;  
    public int mileage;  
}  
public class Plane {  
    public Propeller[] myPropellers;  
    public Engine myEngine;  
}  
public class Pilot {  
    public int flightHours;  
    public void fly(Plane p) {  
        ...  
    }  
}  
public class FighterPilot extends Pilot {  
    public int rank;  
}
```



- n. (4 pts) Consider the UML diagram on the right:
- a. Which class contains class D? **B, F**
 - b. Which class uses class D? **C**
 - c. Which class may change if class D changes? **B, C, F**
 - d. How many instances of class D does class F have? **4**
 - e. Can class A be used wherever Class C is used? **No**
 - f. Can class E be used wherever Class C is used? **Yes**



Problem 4 Java Programming (25 pts)

- o. (4 pts) Java Language Features
- a. Using “==” and .equals() always return the same result **T or F**
 - b. Variables of type Integer and int are both references **T or F**
 - c. Autoboxing creates an Integer object from an int **T or F**
 - d. Exceptions are used to capture run-time errors **T or F**

- p. (4 pts) Write Java code for a Card class
- Use an enumerated type for the suits in a card deck (Spades, Hearts, Diamonds, Clubs)
 - Implement the comparable interface for Card objects so that the suits are ranked in the order listed (Spades > Hearts > Diamonds > Clubs)

```

public class Card implements Comparable {
    public enum Suit {Spades, Hearts, Diamonds, Clubs}
    private Suit mySuit;
    Card (Suit s)           { mySuit = s; }
    Card (int i) {
        if (i == 1) mySuit = Suit.Spades;
        else if (i == 2) mySuit = Suit.Hearts;
        else if (i == 3) mySuit = Suit.Diamonds;
        else mySuit = Suit.Clubs;
    }
    public Suit getSuit() { return mySuit; }
    public String toString(){ return mySuit.toString(); }

    public int compareTo(Object o) { // return 1 if greater, 0 if equal, -1 if less than
        Card card = (Card) o;
        Suit c = card.getSuit();
        if (mySuit == Suit.Spades) {
            if (c == Suit.Spades) return 0;
            else return 1;
        } else if (mySuit == Suit.Hearts) {
            if (c == Suit.Spades) return -1;
            else if (c == Suit.Hearts) return 0;
            else return 1;
        } else if (mySuit == Suit.Diamonds) {
            if ((c == Suit.Spades) || (c == Suit.Hearts)) return -1;
            else if (c == Suit.Diamonds) return 0;
            else return 1;
        } else // (mySuit == Suit.Clubs) {
            if (c == Suit.Clubs) return 0;
            else return -1;
        }
    }
    public static void main(String[] args) { // prints 1 0 -1
        Card s = new Card(Suit.Spades);
        Card h = new Card(Suit.Hearts);
        System.out.println("S vs H = "+s.compareTo(h));
        System.out.println("S vs S = "+s.compareTo(s));
        System.out.println("H vs S = "+h.compareTo(s));
    }
}

```

- q. (4 pts) Write Java code for a Deck class
- Uses an ArrayList to store multiple Card objects
 - Use an anonymous inner class to generate an Iterator over Card objects in the Deck

```
public class Deck {
    private ArrayList myCards;
    Deck () { myCards = new ArrayList(); }
    public void addCard(Card c) { myCards.add(c); }

    public Iterator iterator() {
        return new Iterator() { // unnamed inner class implementing Iterator
            private int pos = 0;
            public boolean hasNext() { return (pos < myCards.size()); }
            public Object next() { return myCards.get(pos++); }
            public void remove() {}
        };
    }

    public static void main(String[] args) { // prints Spades Clubs Hearts
        Deck d = new Deck();
        d.addCard(new Card(1));
        d.addCard(new Card(4));
        d.addCard(new Card(2));
        Iterator it = d.iterator();
        while (it.hasNext()) {
            System.out.println(it.next());
        }
    }
}
```

Problem 5 Graphic User Interfaces (15 pts)

- r. (4 pts) In a graphics user interface
- What is the model? **The application and its data**
 - What is the view? **The visual representation of the model**
 - What is the controller? **Controls interactions with the user**
 - Why should these be kept separate?
Separate data from its appearance, more robust, easier to maintain
- s. (4 pts) Event driven programming
- What are events?
Action or condition occurring outside normal flow of control of program
 - Why use events for GUIs?

Because we cannot predict the time and occurrence of an event

c. How are events handled in Java Swing?

Create and register an ActionListener object for each event, Java will invoke a method in the ActionListener whenever the event occurs

t. (4 pts) Given Java Swing code for a GUI, understand the different roles of each part

```
import java.awt.Dimension;
import javax.swing.*;
```

```
public class Application extends JPanel {
    private static final int FRAME_WIDTH = 300;
    private static final int FRAME_HEIGHT = 400;
    private static final int XCOORD = 100;
    private static final int YCOORD = 200;
    private JFrame frame;
    private String title = "Second Example";

    public Application() {
        // Creates Java frame object with desired dimensions
        frame = new JFrame();
        frame.setContentPane(this);
        frame.setLocation(XCOORD, YCOORD);
        frame.setSize(new Dimension(FRAME_WIDTH, FRAME_HEIGHT));
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.setTitle(title);
        frame.setVisible(true);

        // MouseListener class will take care of mouse events
        addMouseListener(new MouseHandler());
    }

    public static void main(String[] args) {
        new Application();
    }
}

public class MouseHandler implements MouseListener {

    public void mouseClicked(MouseEvent e) { // Handler called when event occurs

        // Performs activity to handle event
    }
    ...
}
```