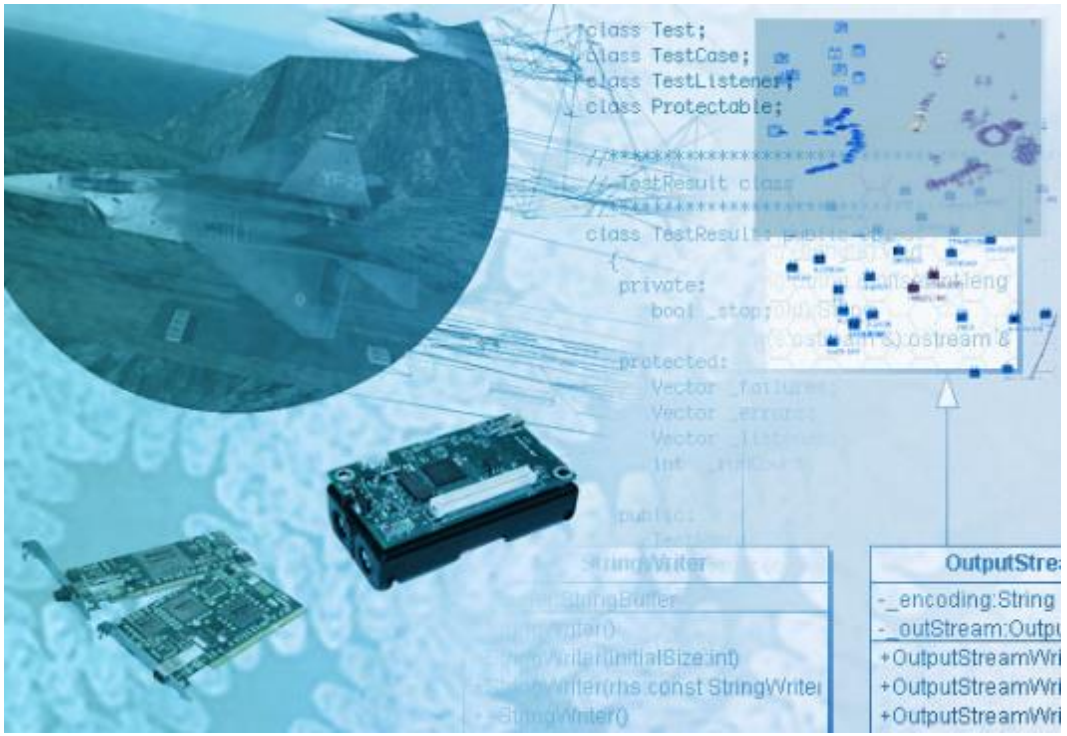
CSYE 6200

CONCEPTS OF OBJECT-ORIENTED DESIGN

SESSION 9



MARK G. MUNSON

ADMINISTRATION

- **Mar 22nd – Session 9: Inner and Anonymous Classes**
  - Enumeration
  - Design Pattern: Command
  - UML: Sequence Diagram
  - Assignment #5a target due date (not collected)
- **Mar 29th – Session 10: Communication/Networking**
  - Design Pattern: Abstract Factory
  - HTTP Networking
  - Lambdas
  - Assignment 5b diagram review
- **Apr 5th – Session 11: Complex Development/Deployment**
  - Libraries and Packages
  - GUI / GIS Development
  - Quiz 3
- **Apr 12th – Session 12: Extra**
  - Assignment $5abc due
- **Apr 19th – Session 13: Full Review**
- **Apr 26th – Final Exam (Online proctored)**

THE LECTURE

- **Recap**
- **Enumeration**
- **Java Keywords**
- **Design Pattern**
  - Command
- **Classes**
  - Inner
  - Anonymous
- **UML**
  - Use Cases / User Stories
  - Sequence Diagram
- **Assign 5abc UI Development**

RECAP

ENUMERATION

ENUMERATION

Enumeration constants may also be defined to accept constructor input values

```
enum Weekday { SUNDAY(0), MONDAY(1), TUESDAY(2),
WEDNESDAY(3), THURSDAY(4), FRIDAY(5), SATURDAY(6);
  private int n; // member variable
  Weekday(int n) { val = n; }  // constructor
…
}
…
for (Weekday day : Weekday.values() ) {
  System.out.println(day + " " + day.ordinal());
 }
```

ENUMERATION DEMO

JAVA

KEYWORDS

JAVA KEYWORDS

abstract	assert	boolean	break	byte	case
catch	char	class	const	continue	default
do	double	else	enum	extends	final
finally	float	for	goto	if	implements
import	instanceof	int	interface	long	native
new	package	private	protected	public	return
short	static	stictfp	super	switch	synchronized
this	throw	throws	transient	try	void
volatile	while				

- **During this course, we've used nearly every Java keyword that is available.**
- **Only a few special cases remain unused.**

DESIGN PATTERN

COMMAND

DESIGN PATTERNS

- **Creation Patterns**
  - Singleton
- **Behavioral Patterns**
  - Observer
  - Command

DESIGN PATTERN: COMMAND

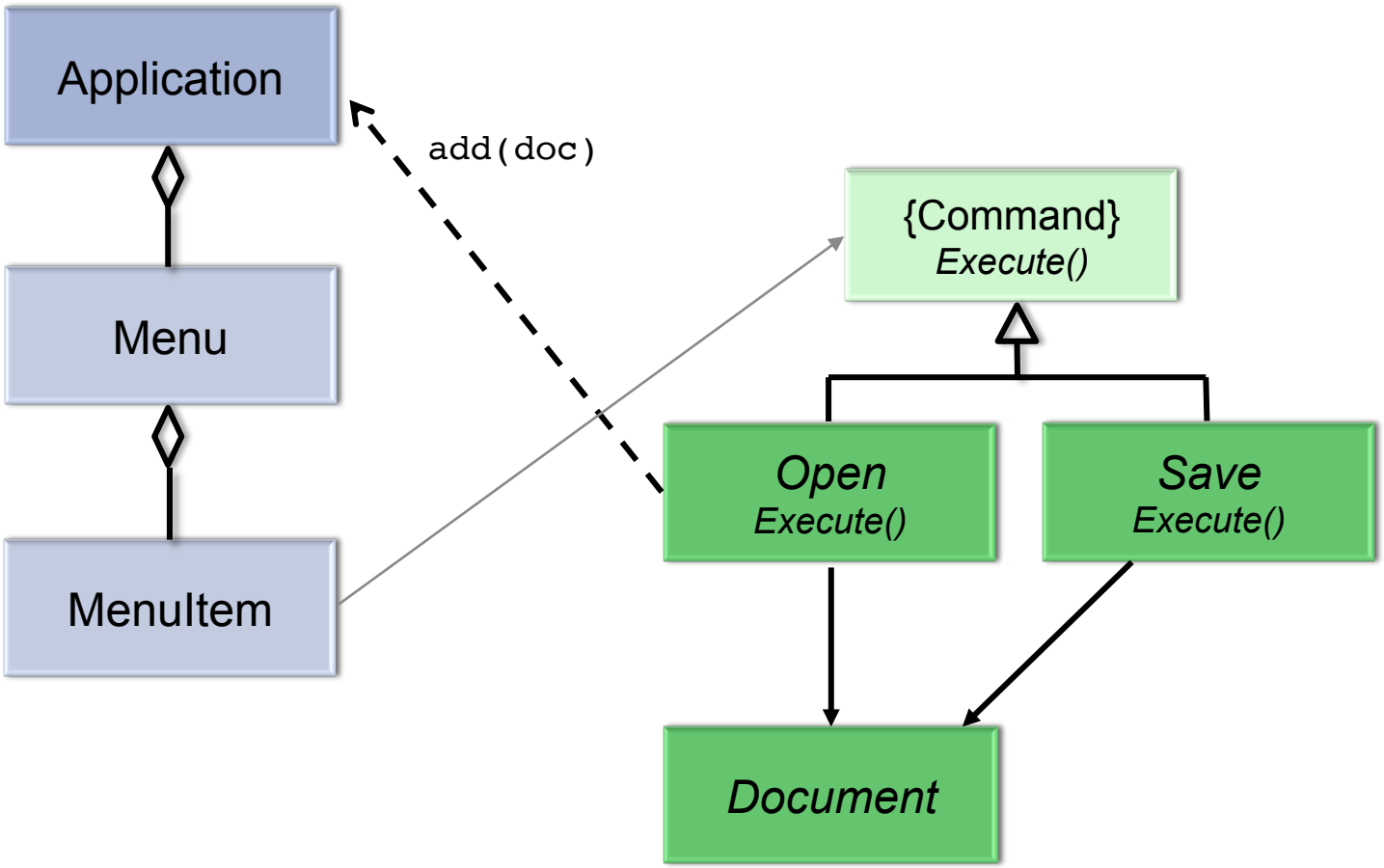
- A system often want to issue requests without knowing anything about the resulting operation.

- The Command pattern lets a system make requests by turning the command itself into an object.

- The primary aspect of this pattern is an abstract command class, which provides an interface for executing operations.

```
can easily undo/redo
```

COMMAND



COMMAND DEMO

A.K.A. NESTED CLASSES

INNER CLASSES

INNER CLASS

```
public class InnerTest {
   private int key = 20;

   public InnerTest(int key) {
    this.key = key;
   }

   public class InnerCls {
      public String encode(String input) {
      StringBuffer sb = new StringBuffer();
      for (char c : input.toCharArray()) {
        char c2 = c ^ key;
        sb.append(c2);
        }
      return sb.toString();
      }
 }


public InnerTest() {
   InnerCls iproc0 = new InnerCls();
```

communication mechanism

The Inner class can see private member variables of the outer class

INNER DEMO

ANONYMOUS INNER CLASS

ANONYMOUS INNER CLASS

- **Often in User Interface applications, you wish to create a class with three properties:**
  1. It's used once for a unique purpose
  2. Usually with a single defined method (i.e. actionPerformed(), run(), etc.)
  3. The class should have access to local member variables
- **Anonymous Inner Classes permit you to create a class 'on the fly' without giving it a formal name**

```
callback for subscription
```

ANONYMOUS INNER

```
/*
 * Start a timer task to periodically redraw the UI display panel
 */
private void startPanelTimerTask() {
        Timer timer = new Timer();
        timer.scheduleAtFixedRate(new TimerTask() {

            public void run() {
              panel.repaint();
            }
        }, 0, 1000); // update every 1000 ms (second)
}
```

ANONYMOUS INNER

```
btn0 = new JButton("Start");
//btn0.addActionListener(this);

// Anonymous Inner Class
btn0.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        startSim();
    }
});
```

UI JPANEL

USE CASES

USE CASES / USER STORIES

- **Use Case provide the following benefits**
  1. Provide a clear description of what the system will do
  2. Help to describe the functional requirements of the system
  3. Help to establish testing by identifying verification and validation opportunities
  4. Provide a path for defining classes and objects

USE CASE NAME/DESCRIPTION

Use Case: *<Use case name>*
Id:
Level: *<Low, Medium, High>*

Description
<A description of what is happening in this use case.>

Actor(s)
<Who are the actors in this use case?>

Stakeholders and Interests
<Who would be affected or interested in this use case?>

USE CASE
MAIN SCENARIO

Pre-Conditions

- *Condition 1*
- *Condition 2*

Trigger

<What happened to start this use case?>

Post-Conditions

Success end condition

Failure end condition

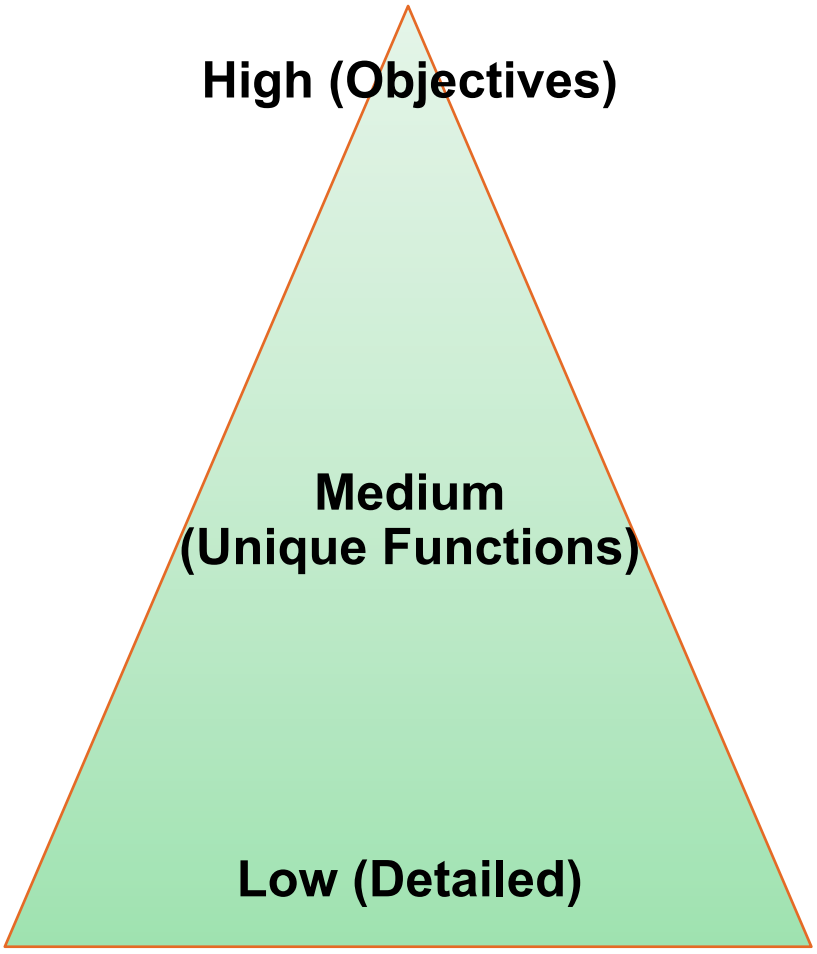
Minimal Guarantee

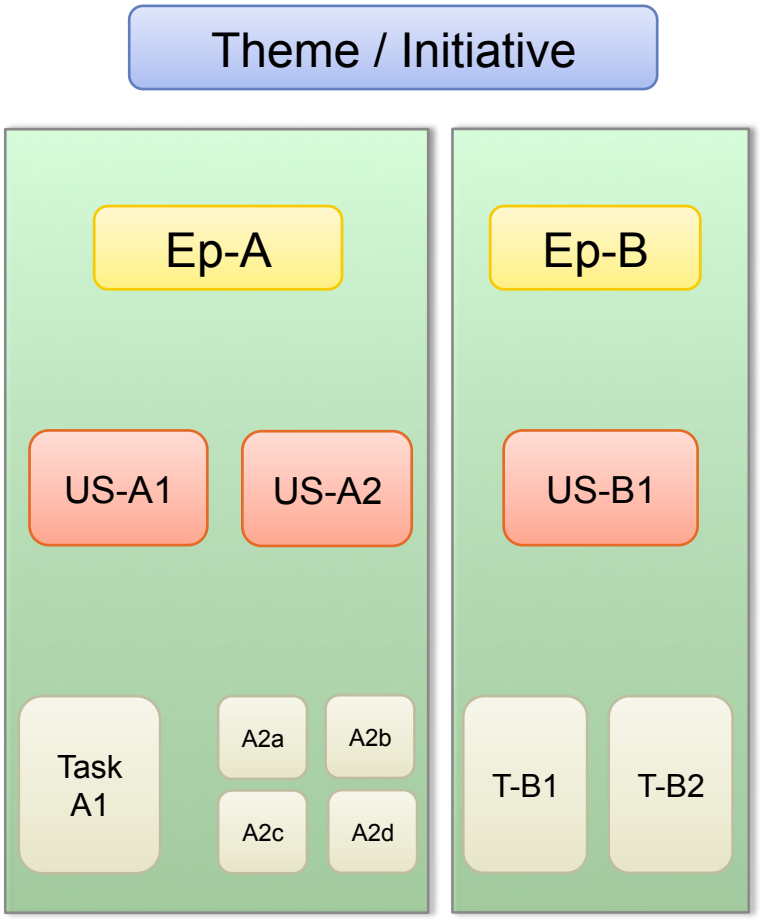
Main Scenario

1. *Step 1*
2. *Step 2*

Alternate Scenarios

UC LEVELS VS. AGILE



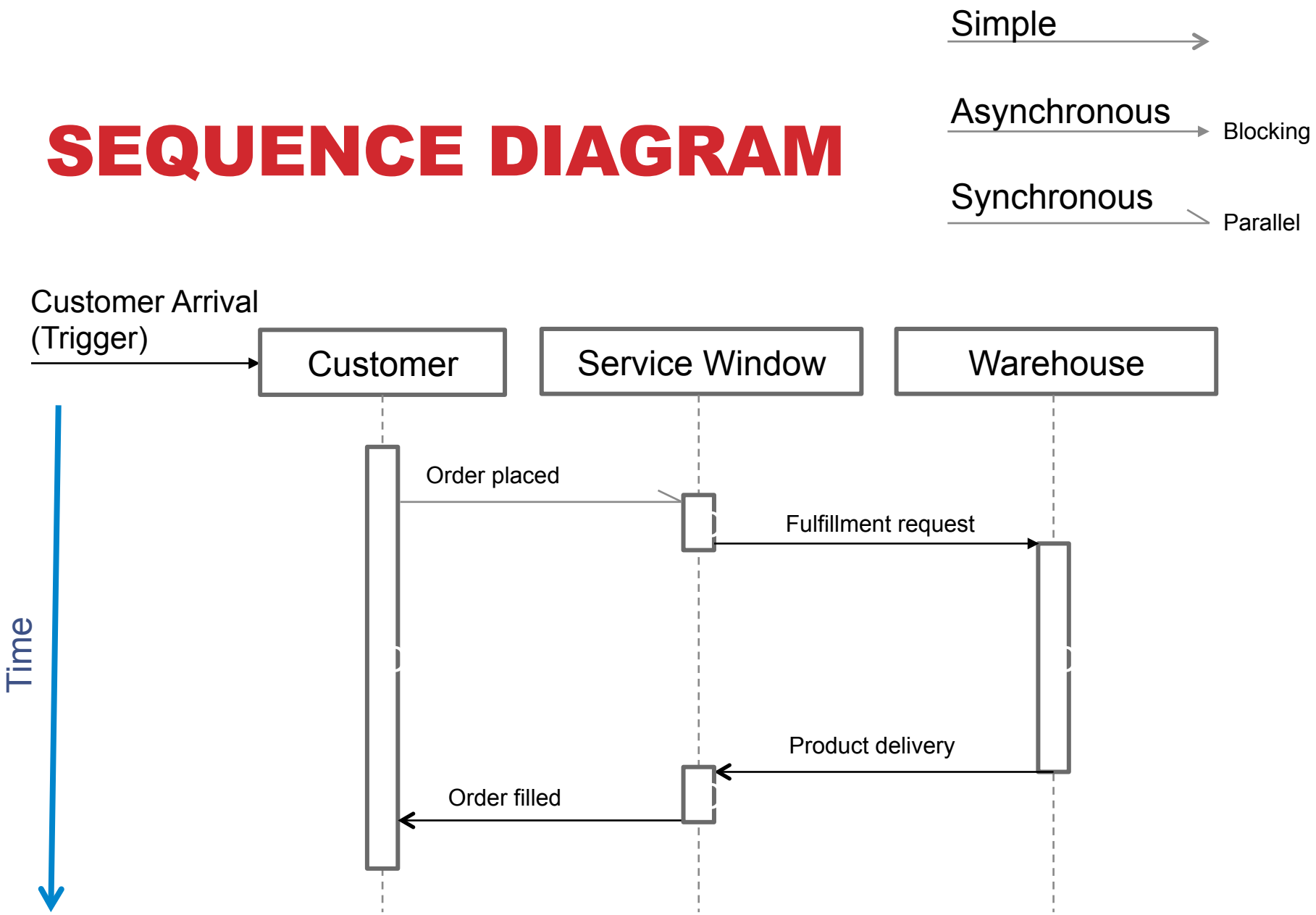


SEQUENCE DIAGRAM

SEQUENCE DIAGRAM

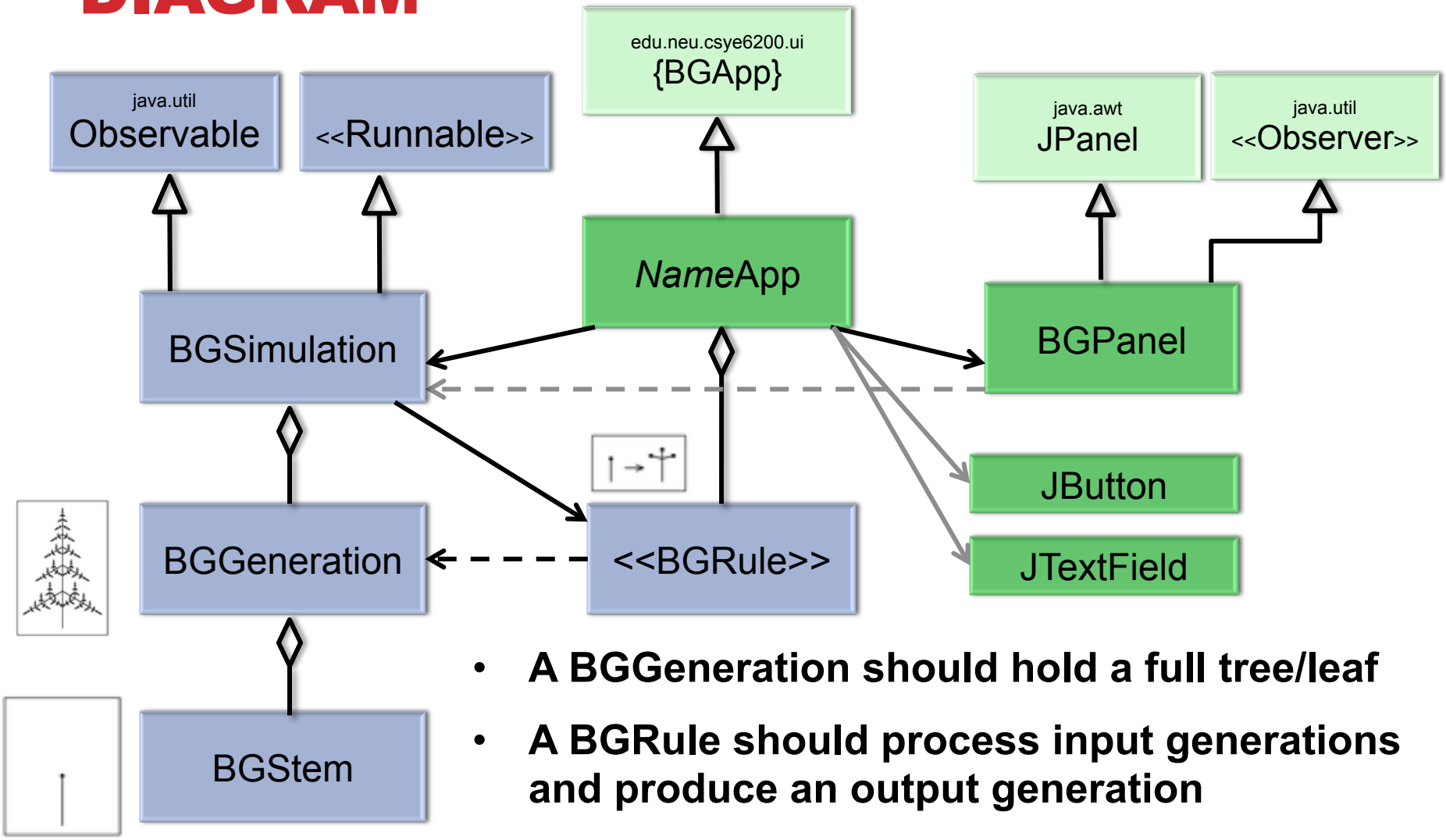
- Class Diagrams are useful for defining the relationships between objects, but they fail to capture the event sequences as described in a Use Case.
- Sequence Diagrams are a way of showing interactions between system elements, with an emphasis on the work being performed.
- Sequence diagrams are especially useful in system that are driven by message exchanges.

SEQUENCE DIAGRAM



BG REVIEW

BG SIMULATION STATIC CLASS DIAGRAM



- **A BGGeneration should hold a full tree/leaf**
- **A BGRule should process input generations and produce an output generation**

NEXT WEEK

JABG Review

- Ch. 14: Lambdas pp. 489-494

Design Pattern: Abstract Factory

- **Continue with Assignment 5abc**
- **Assignment 5b (Diagrams) – Perform this week and next – Review with TA or Instructor**
  - Select a Use Case that explains a significant user operation in your BG application
  - Draw a sequence diagrams for your BG application which corresponds to your selected use case
    - Choose an operation to diagram (startup, rule selection, execution, early termination, etc.)
    - Show User actions or triggers
    - Show major classes or areas of responsibility (i.e. BG Simulation, Rendering, etc.)
    - ***Draw the diagram yourself. Don't use an advanced software tool.***
- **Place name and NUID on the drawings**
  Confer with your TA or Instructor to leverage online office hours
- **NOTE: Include a copy of your documentation with your project submission**