

CMSC 131: Chapter 9 (Supplement)

System Design

Computing and Programs

Review:

- **Software lifecycle**
- **Incremental (stepwise) design**
- **Pseudocode and flowcharting**
- **Prototyping and testing**

Today we take a **wider view** of program development.

Two principal aspects of program design:

A single computational entity:

Coordinating a community of entities:

A Single Computational Entity

Consider a **single computational entity** to solve a specific problem. The method used to solve the problem is called an **algorithm**.

Example: Make a peanut butter and jelly sandwich:

- Get a loaf of bread
- Remove two slices
- Get a jar of peanut butter
- Get a knife
- Open the jar
- Using the knife, get some peanut butter and spread it on one slice
- ...blah, blah, blah

There is essentially **one sequential process** being described.

System Design: What is it?

System Design: Is concerned with coordinating a community of computational entities to achieve a complex process.

Single entity ↔ **Make a sandwich**
System design ↔ **Run a restaurant**

Running a restaurant involves the **coordinated interaction** of many entities:

- **Owner**
- **Chefs**
- **Waiters**
- **Diners**

System Design: What is it?

System Design: Identifying entities, assigning responsibilities, defining how these entities act and interact with each other.

Other Examples of Systems:

Classroom environment: Lecturers, TAs, students, ...

Library: Circulation (checkout and return), indexing services (online catalogue), library users, book buyers, shelvees, ...

Pharmacy: Patients (and medical records), pharmacists, doctors, drug retailers, the pharmacy (products in stock), ...

Video game: Race cars, motorcycles, warriors, space ships, death squads, monsters, aliens, mutants, guns, swords, weapons of mass destruction, cute Japanese cartoon animals with huge eyes, ...

Essential Questions

Challenges:

Essential Questions:

- What is the **desired behavior** of the program (as a whole)?
- What are the **entities** that produce this behavior?
- How does each one **work**?
- How do these entities **interact**?

Behavior

Specifying Desired Behavior: A **use case** is a description of the interaction of a user and the system. It includes:

- **Prerequisites (pre-conditions):**
- **Possible actions and interactions:**
- **Effects (post-conditions):**

Example: Customer in a restaurant.

- **Pre-conditions:**
Customer: hungry and has money
Restaurant: has food
- **Actions:** get menu, order food, be served, eat, pay, leave
- **Post-conditions:**
Customer: less hungry and less money
Restaurant: more money and less food.

Principal Design Elements

Components:

- What are the entities that make up our system?
- What are the roles they play?
- How do we separate the system into distinct units?

Contract: What are the responsibilities and services associated with each component? What guarantees does it make?

State: What is the current status/state of the units that define our system?

Communication: How components request interactions with each other?

Example: Pharmacy Store System

Components: Pharmacist, customers, doctors, prescription, store stock.

Fill-prescription Contract: A valid prescription is presented by the customer. Check patient records and inform of possible side-effects. Dispense the prescription. Update patient records. Deliver medication to patient.

State: For a patient: Current prescriptions, number of times refilled, date of last refill, health insurance information.

Relationship to Java

System: Java program

Components (or community members): Java class objects

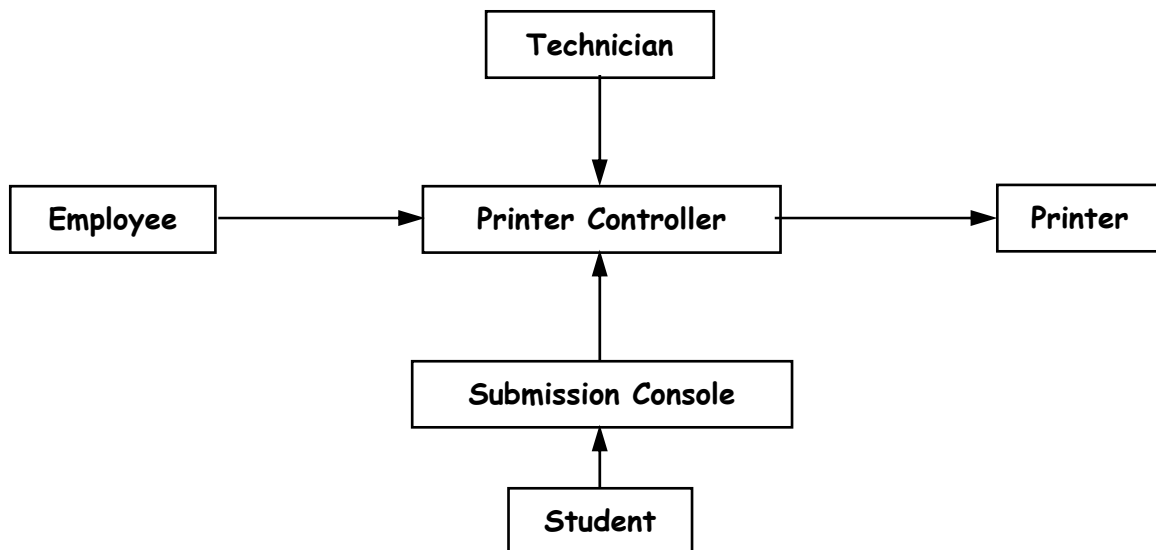
State: Stored in class instance variables.

Contract (or specification): This is called an **API** (Application Programmer Interface), or simply an **interface**. This is the **external** (class user) **view** of an object.

The contract is implemented by the object's class **methods**.

Printer Controller Example

Printer System: A printer system used by students to print projects, homeworks, etc. Students enter a room where the printer resides. Students submit requests from a special console. An employee operates a printer controller, and hands out a print job submitted by students. A technician maintains the printer controller if it breaks.



Printer Controller Example

Use Case Example 1: A student prints a document

Pre-conditions (prerequisites): A **document exist** and is ready to be printed.

Actions:

- if (there is **space available** in the printer queue)
- specify **name of document** to print through the console
 - printer controller accesses document and **sends it to the printer**
- else
- printing of the document **does not** occur
 - an appropriate **error message** is generated on the console

Post-conditions (effects):

- a document has been printed
- or -
- an error message has been generated.

Printer Controller Example

Use Case Example 2: A technician fixes a printer problem

Pre-conditions (prerequisites): A **problem** has been identified in the printer controller.

Actions:

- A technician enters his **password** in the printer controller (to gain access to the system).
- The technician asks for a **status report** from the controller.
- The nature of the **problem is identified**.
- The technician proceeds to **repair** the problem area in the printer.

Post-conditions (effects): The printer is **now operational**. The controller reflects the new status.

Printer Controller Example

Use Case Example 3: An employee picks up a student print-out

Pre-conditions (prerequisites): A document has been sent to the printer by a student through the console.

Actions:

- a student inquires about his/her particular document
- the employee checks the stack of printed documents in the printer tray.
- if document is found
 - the document is handed to the student
- else
 - student is informed that the document is not printed

Post-conditions (effects): A student has a printed document or has been informed that the document is not been printed.

Printer Controller Example: Components

Examples of some Components:

Name: Printer

Description: Provide paper printouts of documents.

State: Paper quantity and other printer status information.

Name: Printer Controller

Description: Used by the Employee and Technician to interact with printer.

State: Must keep track of documents to be processed documents already finished, and status of the printer.

Name: Printer Request Console

Description: Used by students to submit a document to print.

State: The set of user's allowed in the system, and the set of students that are currently in the system.

Printer Controller Example: Interactions

Examples of some Interactions:

Interaction 1: Controller and Printer

The controller sends documents to process to the printer. It also gathers status information from the printer which is made available to the employee or technician.

Interaction 2: Console and Controller

The console sends a query to the controller which determines whether the document can be printed or not. It schedules the job to be printed. Status information about the job is sent back to the console.