

CS2704

Topic:
An Example Design

Outline

- Problem Statement
- System Features
- Class Identification
- Class Relationships
- Scenarios
- Implementation

CS2704 Example Design

2

Problem Statement

A manufacturer of custom bicycles wants a system that allows its customers to place and track orders via the web, its sales staff to manage the orders, its stock manager to manage available components, and its production staff to fill the orders. Bicycle frames are available in many sizes and styles. The customer can also choose the components for the bicycle, but since components may not be available when the order is finally filled the system needs to keep track of suitable replacements.

CS2704 Example Design

3

System Features

- Customer can place order
- Customer can track orders
- Sales can *manage* orders
 - Billing
 - Update status information
 - Shipping

CS2704 Example Design

4

System Features (2)

- Management of inventory of components and frame materials
- Identification of replacement components
- Production staff to fill orders

CS2704 Example Design

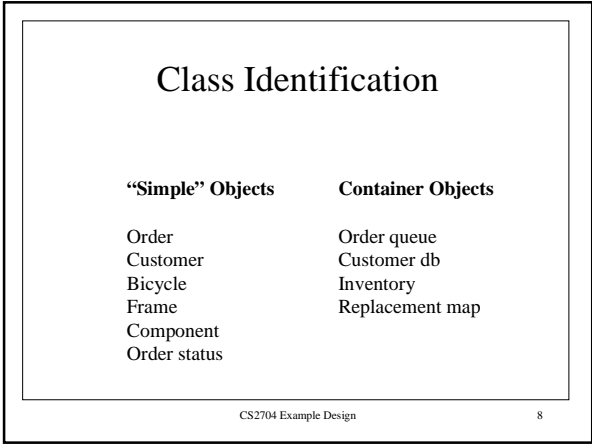
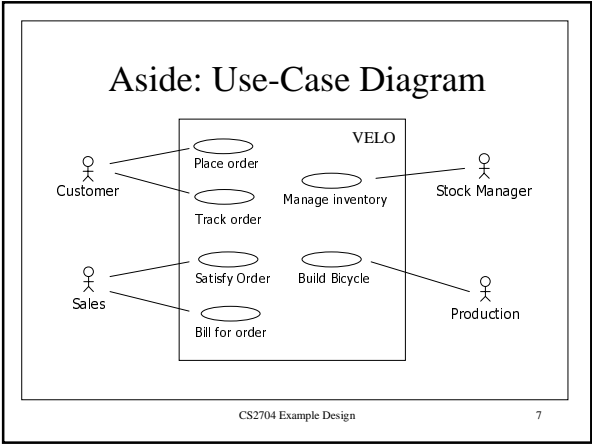
5

User Views of System

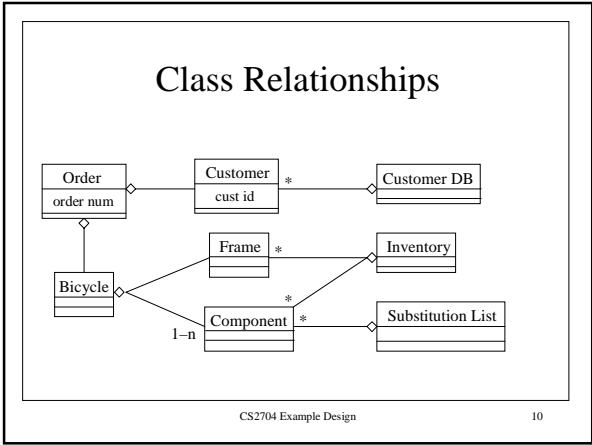
- Customer view: place & track orders
- Sales office: satisfying order, collecting \$
- Stock person: managing inventory
- Production staff: building bicycle

CS2704 Example Design

6



- ### Class Fields
- Order: customer, bicycle, status
 - Customer: name, address, billing Info
 - Bicycle: frame, components
 - Frame: style, size, material, color
 - Frame size: dimensions
 - Component: part name, manufacturer, etc.
- CS2704 Example Design 9

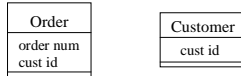


- ### Rethinking Relationships
- Does an Order really have a Customer?
 - More like Customer has Order(s)
 - Don't want to have to maintain
 - multiple copies of order objects
 - pointers to order objects
 - Use order numbers and customer ids to cross reference
- CS2704 Example Design 11

- ### Using Cross-Referencing
- Alternatives:
 - Customer stores list of order numbers
 - Order database is indexed by customer id and order number, Customer has no information about order
 - Order stores customer id in both alternatives
 - Note: Similar issue in Bicycle class
- CS2704 Example Design 12

Rethinking Conclusion

- Lose direct relationship in diagram



- Must have 2-way association between Customer and Order DBs

CS2704 Example Design

13

Order Class Form

- Fields: Order number, Bicycle, Customer, Status
- Operations
 - Accessors:
getOrderNum, getBicycle, getCustomer, getStatus
 - Mutators:
setBicycle, setCustomer, setStatus

CS2704 Example Design

14

Order Class Declaration

```
class Order {
// methods will go here
private:
ord_num orderno; //unique
Bicycle bike_spec; //only one, could be many
cust_id cust; //one customer per order
status_type currstatus;
Order(); //no orders without order numbers
};
```

CS2704 Example Design

15

Order Class Declaration (2)

```
class Order {
public:
Order(const ord_num&);
Order(const ord_num&, const Bicycle&, const cust_id&);
ord_num getOrderNum() const;
const Bicycle& getBicycle() const;
cust_id getCustomer() const;
status_type getStatus() const;
void setBicycle(const Bicycle&);
void setCustomer(const cust_id&);
void setStatus(const status_type&);
private:
// fields go here
};
```

CS2704 Example Design

16

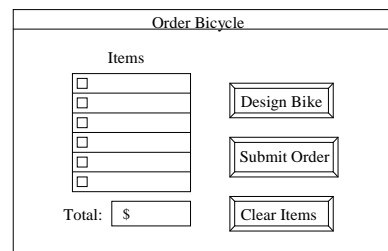
Order Numbers

- Order class on previous slide shows how to set order numbers from outside.
- How could we set order numbers from “inside” class? (hint: static variables)
- What about persistence? (keeping values between executions)

CS2704 Example Design

17

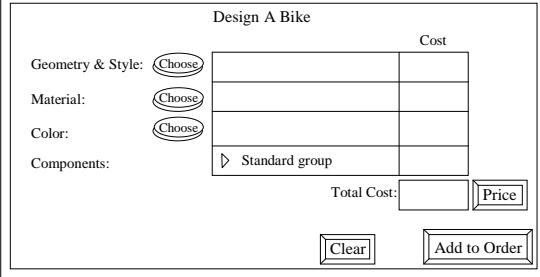
User Interface for Orders



CS2704 Example Design

18

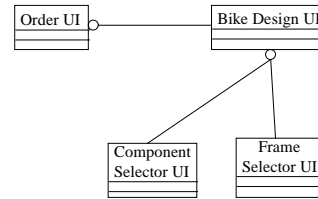
User Interface for Bike Design



CS2704 Example Design

19

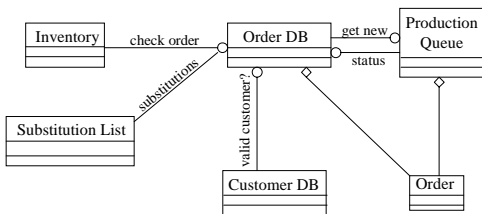
User Interface Classes



CS2704 Example Design

20

Class Relationships (2)



CS2704 Example Design

21

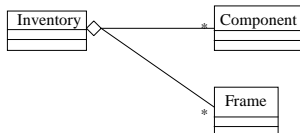
What Happened to the Names?

- Where did the old names go?
- As designer we can change our mind about how things are structured or named before final document
- Must be careful to document changes, and/or update documents.

CS2704 Example Design

22

Inventory Class



CS2704 Example Design

23

Inventory Class Form

- Fields: collection of components
- Methods
 - Accessors
 - listCompByType(type) – return list
 - search(part_number), count(part_number)
 - Mutators
 - add(component), remove(part_number)
 - reserve(part_number) – return false if cannot

CS2704 Example Design

24

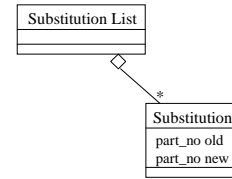
Inventory Class Implementation

```
class Inventory {
public:
    Inventory();
    void add(const Component&);
    void remove(const part_number&);
    bool reserve(const part_number&);
    CompList listCompByType(CompType) const;
    bool search(const part_number&);
    int count(const part_number&);
private:
    //data structure for inventory
};
```

CS2704 Example Design

25

Substitution List



CS2704 Example Design

26

Substitution List Class Form

- Fields: collection of substitutions, Inventory association
- Methods
 - Accessors
 - findSubstitute(part_number)
 - Mutators
 - Add(substitution)
 - Delete(substitution)
 - removeInvalid() – remove and return list of invalid substitutions

CS2704 Example Design

27

Substitution List Class Impl.

```
class SubstitutionList {
public:
    SubstitutionList(Inventory&); //create empty list
    part_number findSubstitute(const part_number&) const;
    ListofSubst removeInvalid(); //need better return type!
    add(const Substitution&);
    delete(const Substitution&);
private:
    //data structure for collection of substitutions
    Inventory* inv;
};
```

CS2704 Example Design

28

Constructor for Subst List

```
SubstitutionList(Inventory& i) : inv(&i) {
    //initializations for data structure
}
```

CS2704 Example Design

29

Order DB Class Form

- Field: Collection of Orders
- Methods:
 - Accessors
 - find(OrderNum), find(Customer)
 - Mutators
 - remove(OrderNum), add(Order)
 - getProductionOrders()
 - Static associations

CS2704 Example Design

30

Order DB Implementation

```
class OrderTable {
public:
    OrderTable(Inventory&, ProductionQueue&, CustomerDB&);
    Order& find(const OrderNum&) const;
    Order& find(const Customer&) const;
    OrderList getProductionOrders(); //return type?
    add(const Order&);
    remove(const OrderNum&);
private:
    //data structure for collection of order objects
    Inventory* inv; ProductionQueue* pque; CustomerDB* custs;
};
```

CS2704 Example Design

31

Order DB Constructor

```
OrderTable(Inventory& i, ProductionQueue& p, CustomerDB& c) :
    inv(&i), pque(&p), custs(&c)
{
    pque->useOrderTable(this); //two way association
    //code to initialize data structure goes here
}
```

CS2704 Example Design

32

Production Queue Class Form

- Field: queue of orders
- Mutator
 - Accessor:
 - listOrders(order_status)
 - getStatus(OrderNum)
 - Mutator:
 - Remove(OrderNum)
 - nextOrder() – get next available order
 - completeOrder(OrderNum)

CS2704 Example Design

33

Production Queue Impl.

```
class ProductionQueue {
public:
    ProductionQueue(); //empty queue
    void useOrderTable(OrderTable&); //association
    OrderList listOrders(order_status) const;
    order_status getStatus(const OrderNum&) const;
    void remove(const OrderNum&);
    const Order& nextOrder();
    void completeOrder(const OrderNum&)
private:
    Queue q;
    OrderTable* orderdb;
};
```

CS2704 Example Design

34

Scenarios

- Will consider:
 - Adding order to order db
 - Getting next order from production queue
 - Completing order in production queue

CS2704 Example Design

35

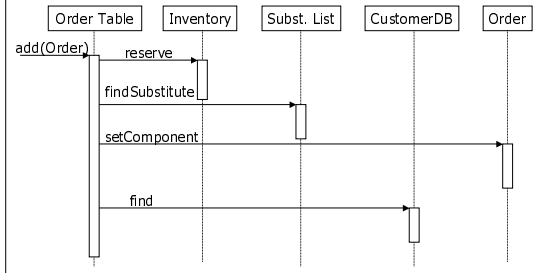
Scenario: Adding Order

- Before adding, check that order can be filled and that customer is in database:
 - For each component
 - if component is not available, find substitute
 - If customer in database then proceed with order

CS2704 Example Design

36

Adding an Order



CS2704 Example Design

37

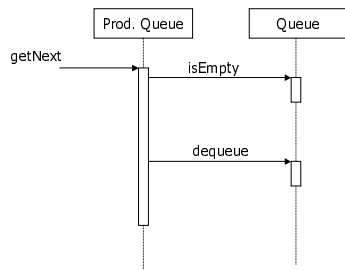
Scenario: Get Next Order

- Logic:
 1. If there is an order not in production, return that order
 2. If there is not, get new orders from Order db, return first
 3. If there are none, return ?
- Found a problem: need to rethink return type, or methods

CS2704 Example Design

38

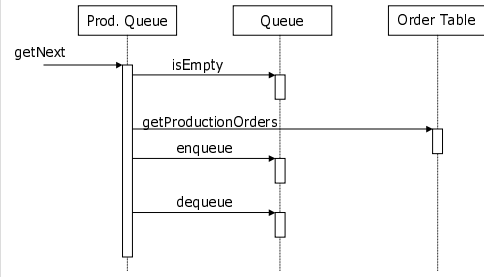
Get Next Order (nonempty)



CS2704 Example Design

39

Get Next Order (Empty Queue)



CS2704 Example Design

40

Next Order Problem

- What to do if there are no new orders in order db?
- Can't keep Order & getNext() as is

CS2704 Example Design

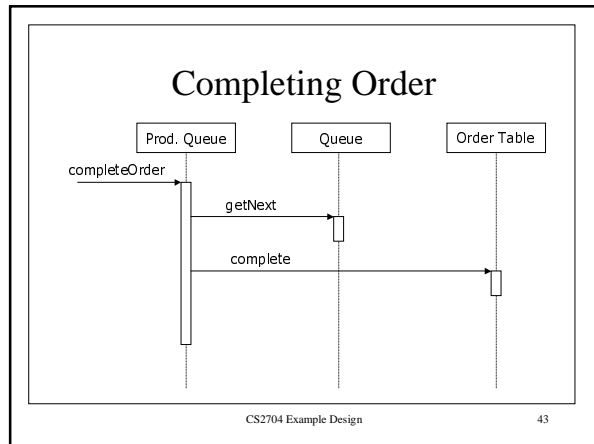
41

Scenario: Completing Order

- Logic:
 1. Remove from queue
 2. Send back to Order DB as "completed"
- Issues:
 - How do we send back?

CS2704 Example Design

42



- ### Other Implementation Issues
- What kind of data structures?
 - Inventory – find by part number
 - Substitution list – find by part number
 - Order db – find by order number and cust name (or maybe an id?)
 - Production queue – find first order not being worked on, find specific orders being worked on
- CS2704 Example Design 44

- ### Data Structures
- “Find” operations: use of index structures
 - binary search trees, hashing, b-trees
 - Standard Template Library has indexed containers `map<>` and `multimap<>`
 - “Find first” implies queue structure
 - Can put off data structure details until later, but can recognize needs as go along
 - Ex.* Production queue really needs two data structures.
- CS2704 Example Design 45

- ### Notes
- Design is not a linear thought process
 - Problems found during design are easier to fix than problems found during coding.
- Moral:** *more* time should be spent on design than coding
- CS2704 Example Design 46