Code

- Cover bases
- Work yourself out of a job

Design possible in many ways

- OptionSet contained in Automobile; not inner class
- Option contained in OptionSet; but inner class

Strategies for reusability

- Layers
- Extensibility

Reading file without a buffer

- Reading as we go along is fast
- But we can't predict what is coming next
- Easy to break such a program

Reading file with buffer

- Slower
- Don't have to follow as many rules
- Easy for user, difficult for programmer

Serialization

- static methods & members cannot be serialized because they don't exist within the scope of an object
- members declared transient are excluded from serialization
- contained classes must also implement Serializable

Serializable (the interface)

- the interface itself doesn't do anything (like necessitate methods that other interfaces do)
- it is basically a flag for the ObjectOutputStream & ObjectInputStream to know what to serialize

Encapsulation

- protected methods in contained classes
- Protected from the outside except through pipelines that we create

Containment

- Objects within objects
- Has-a relationship

Association
- Exposes one object to another in some form
- Uses-a relationship

Inheritance
- Reuse code
- Follow object oriented design
- Understandable to us
- Natural

Polymorphism
- Using parent class to refer to child class
- Group things before they get too specific
- Natural

Interfaces
- Components to “throw” on things
- i.e. attach an extra arm
- Methods are necessitated and need to be implemented
- They can also be inherited

API
- Set of methods
- Use: given to manufacturers for offering options to users
- Doesn’t require programmer to integrate new things every time

Interface as an API
- Easy to modify and add methods when needed
- Helpful for car manufacturers and programmers

Taj Mahal example (design analogy)
- Foyer: abstract class, no boundaries, mess for all spaghetti code
- Bunch of interfaces leading to places

CRUD Operations
- Create

- Read
- Update
- Delete

ProxyAuto/BuildAuto together
- Center of operations
- Where our data really is
- Is only briefly exposed in a controlled environment

ProxyAuto
- Behind closed doors
- Behaviour defined

BuildAuto
- What we can show
- No behaviour, just showing that the API exists

LinkedHashMap
- Basically a hash table
- Except a map
- Linkedlist implementation

Memory
- Split

Heap
- All over the place
- Big

Stack
- System call order
- Method calls

DS
- Data segment
- Determined by code
- Contains data

CS

- Code segment
- Just text with instructions in the order we give them

Framework exposing
- Middle layer can be modified easily without breaking the inner layer
- Complexity of product is hidden and irrelevant

Exception handling
- Catching problems that we know to occur
- Providing useful feedback

Self-healing
- Fixing the problems before things break
- Plan b
- Mostly a fantasy for a program to be fully self-healing

Multithreading
- Task == Thread
- Multitasking, multiprocessing

True multitasking
- CPU1 + CPU2 + … + CPUn

Thread
- A thread lives inside a process(or)

Context Switching
- Switching threads
- Moving between different contexts
- An internal back-and-forth

Syncing Shared Resources
- Accessing a data structure one at a time
- Rather than running at it all at once and risking corruption

Process Context
- Where we currently are in the program
- Where did we come from and where are we going?

Preemptive and non-preemptive complexity on threading implementation in JVM

- Priorities can be assigned
- JVM uses OS priority
- Threads “preempt” one another

Locking for shared resource (and management)

- Resource inaccessible for duration of use
- Shared between threads
- To avoid corruption

Explicit thread control

- We determine what to do in separate threads
- As opposed to what already exists

Hypervisor

- Hardware
- CPU

Object locking (syncing)

- “soft” lock, i.e. does not lock contained objects
- analogy: lungs example

LHM static?

- static because the functions from interface have to instantiate buildauto
- with static object, we can keep using the same object
- less code, time, memory used

debug == true

- executes a bunch of println statements for debugging

Syncing in automobile

- Makes sure we don’t lock the object for too long
- Only for the part where actual modifications are made

What to sync

- Things that can break/create links in objects

Not to sync

- Things like print and get

- Not modifying anything
- No chance of data corruption

Client
- client: client package
- 1st copy of the app

Server
- server: server package
- 2nd copy of the app

Interfaces
- another layer
- easy to modify

Abstract
- Another layer
- Separating what is exposed

Relational database
- Relating stuff in different tables

SQL
- Structured query lang
- Relational db language