REKAYASA PERANGKAT LUNAK

"Pemodelan Analisis"

Ratna Wardani

Pemodelan analisis

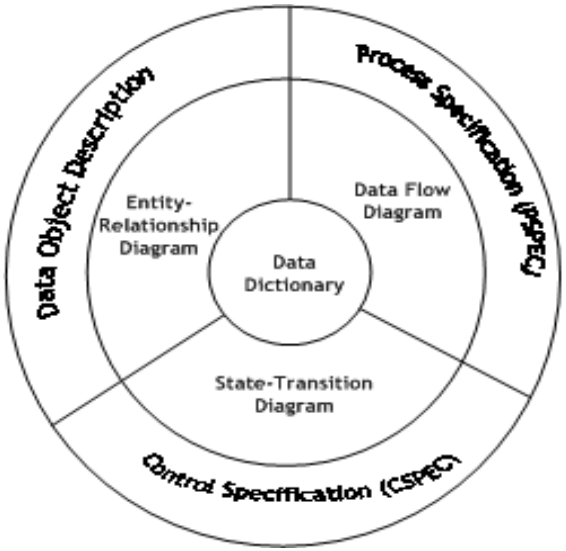
- Rekayasa perangkat lunak dimulai dg serangkaian tugas pemodelan yg membawa pd suatu spesifikasi lengkap dari persyaratan dan representasi desain yg komprehensif bagi S/W yg akan dibangun
- Dua pemodelan analisis : Analisis Terstruktur dan Analisis Berorientasi Objek
- Pemodelan Analisis harus mencapai tiga sasaran utama :
  - menggambarkan apa yg dibutuhkan pelanggan
  - membangun dasar bagi pembuatan desain S/W
  - membatasi persyaratan yg dapat divalidasi begitu S/W dibangun

Elemen Model Analisis

- Struktur Model Analisis (gambar)
- Data Dictionary : deskripsi semua objek data dalam S/W
- Entity Relationship Diagram : notasi pemodelan data yang menggambarkan hubungan antar objek data
- Data Flow Diagram : model fungsional, dengan tujuan
  - menunjukkan transformasi data saat data bergerak melalui sistem
  - menunjukkan fungsi-fungsi yg mentransformasi aliran data
- State Transition Diagram : model tingkah laku, yg menunjukkan transisi state/tingkah laku sistem akibat kejadian eksternal

Elemen Model Analisis

- Struktur Model Analisis

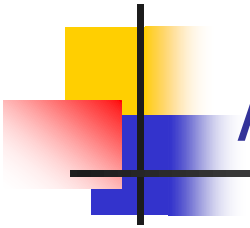


Pemodelan Data

- ERD memungkinkan perekayasa S/W mengidentifikasi objek data dan hubungannya menggunakan notasi grafis (data yg dimasukkan, disimpan, ditransformasi dan dihasilkan suatu aplikasi)
- ERD hanya berfokus pada data dan bersifat independen thd proses yg mentransformasikan data tersebut
- Model data terdiri dari tiga informasi utama :
  - Objek data
  - Atribut
  - Hubungan

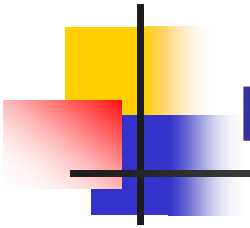
Objek Data

- Objek data adalah representasi dari hampir semua informasi gabungan yg harus dipahami perangkat lunak
- Objek data dapat berupa entitas eksternal, benda, event, unit organisasional, tempat atau suatu struktur
- Deskripsi objek data menghubungkan objek data dg semua atributnya
- Objek data dihubungkan satu sama lain berdasarkan konteks masalah yg dianalisis
- Objek data hanya mengenkapsulasi data, tidak ada referensi pd sebuah objek data ke operasi yg bekerja pada data



Atribut

- Atribut menentukan properti suatu objek data
- Atribut digunakan untuk
  - menamai sebuah contoh dari objek data
  - Menggambarkan contoh
  - Membuat referensi ke contoh lain pada tabel yang lain
- Contoh : objek data **manusia**, memiliki atribut : **nama**, **alamat**, **umur**, **tinggi badan**.
- Rangkaian atribut yang sesuai untuk suatu objek data ditentukan melalui pemahaman konteks masalah



Hubungan

- Objek data dihubungkan satu dan lainnya dengan berbagai cara
- Contoh : Antara dua objek data **buku** dan **toko buku** dapat dibangun suatu hubungan berdasarkan konteks perangkat lunak yg akan dibangun (dg *object-relationship pairs*) :
  - toko buku memesan buku
  - toko buku menampilkan buku
  - toko buku menjual buku
  - toko buku mengembalikan buku

Kardinalitas

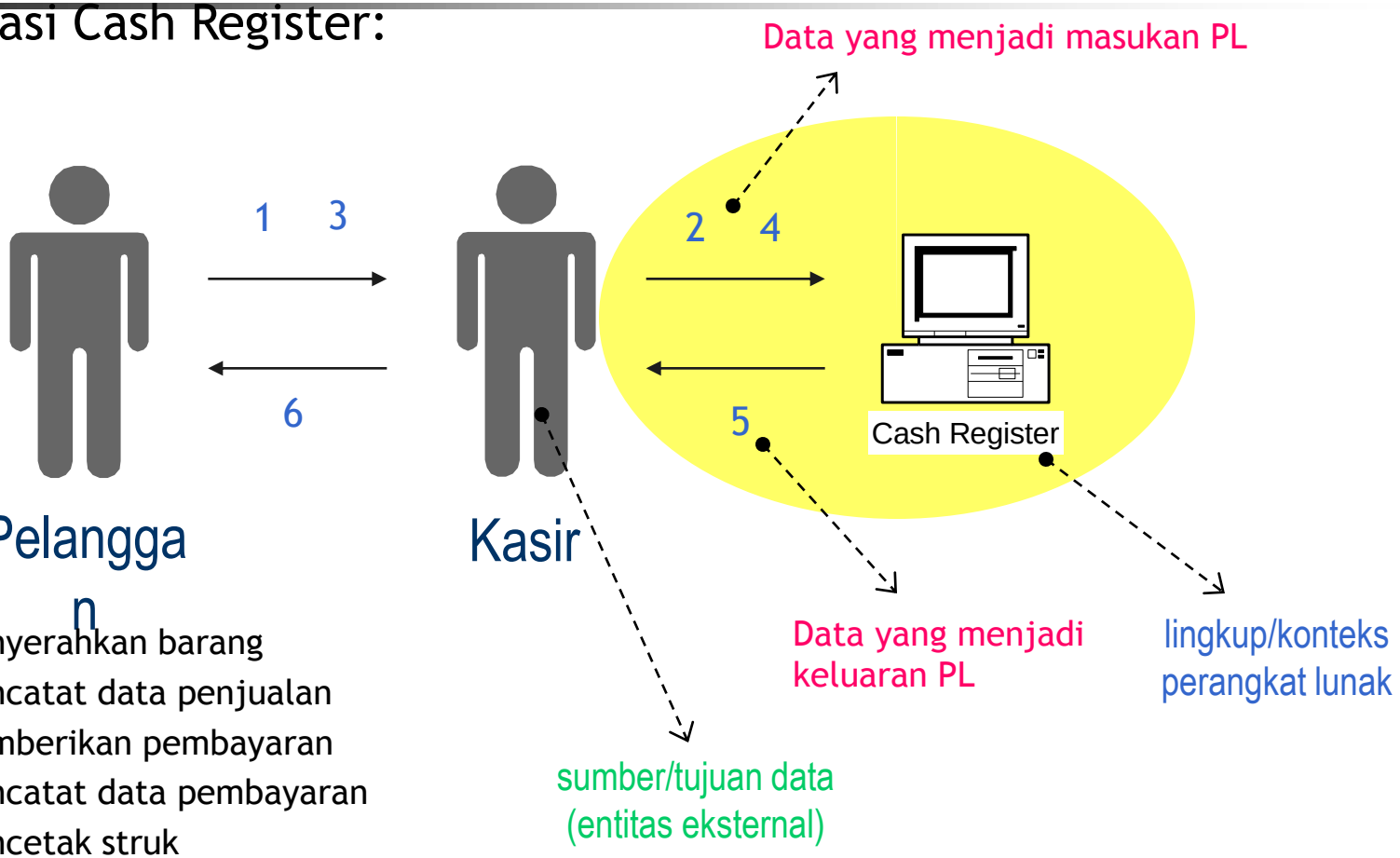
- Kardinalitas merupakan spesifikasi dari sejumlah peristiwa dari satu objek yg dapat dihubungkan ke sejumlah peristiwa dari objek lain
- Dua objek dapat dihubungkan sebagai :
  - Satu-ke-satu : suatu kejadian dari objek A dapat berhubungan dg satu dan hanya satu kejadian dari objek B dan sebaliknya
  - Satu-ke-banyak : satu kejadian dari objek A dapat berhubungan dg satu atau lebih kejadian dari objek B, tetapi satu kejadian dari objek B dapat berhubungan dg hanya satu kejadian dari objek B
  - Banyak-ke-banyak : sebuah kejadian dari objek A dapat berhubungan dg satu atau lebih kejadian dari objek B, dan satu kejadian dari objek B dapat berhubungan dg satu atau lebih kejadian dari objek A

Pemodelan Fungsional dan Aliran Informasi

- DFD merupakan teknik grafis yang menggambarkan aliran informasi dan transformasi yang diaplikasikan pada saat data bergerak dari input menjadi output.
- DFD memberikan suatu mekanisme bagi pemodelan fungsional dan pemodelan aliran data
- DFD dapat menyajikan perangkat lunak pada setiap tingkat abstraksi, karena DFD dapat dipartisi ke dalam tingkat-tingkat yang merepresentasikan aliran informasi yang bertambah dan fungsi ideal.

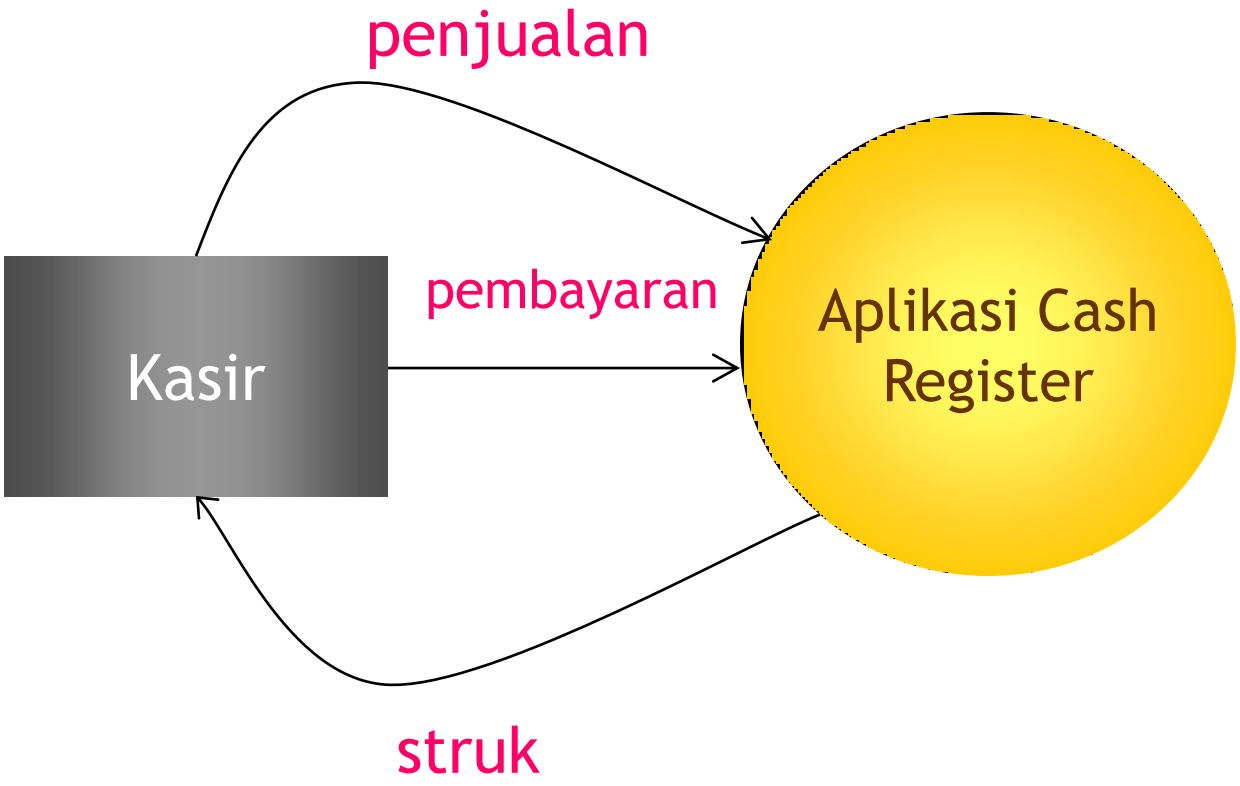
Pemodelan Fungsional dan Aliran Informasi (Analisis Terstruktur)

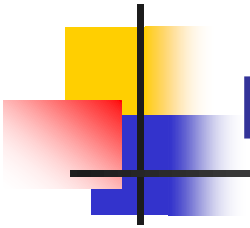
Aplikasi Cash Register:



1. Menyerahkan barang
2. Mencatat data penjualan
3. Memberikan pembayaran
4. Mencatat data pembayaran
5. Mencetak struk
6. Menerima struk, barang, dan kembalian

Pemodelan Fungsional dan Aliran Informasi (Analisis Terstruktur)

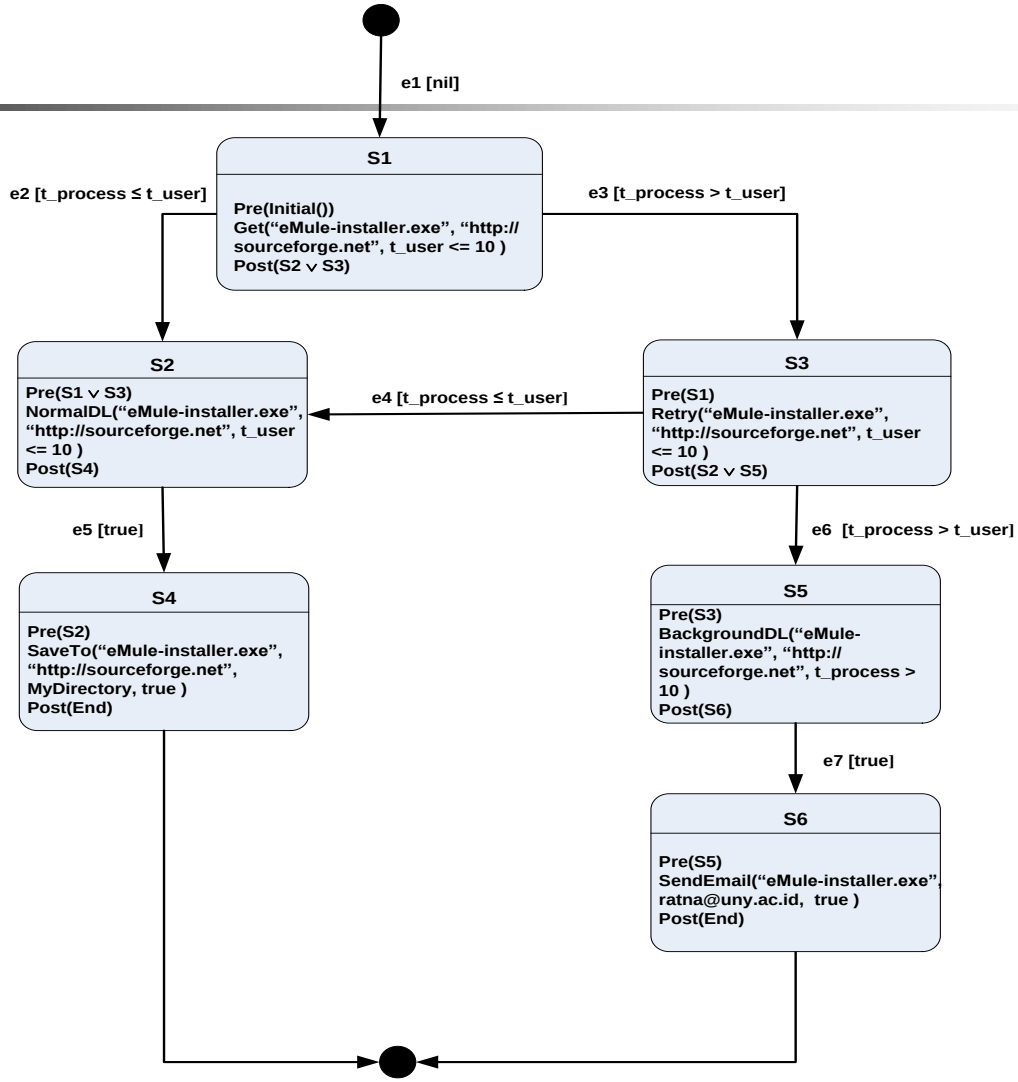




Pemodelan Tingkah Laku

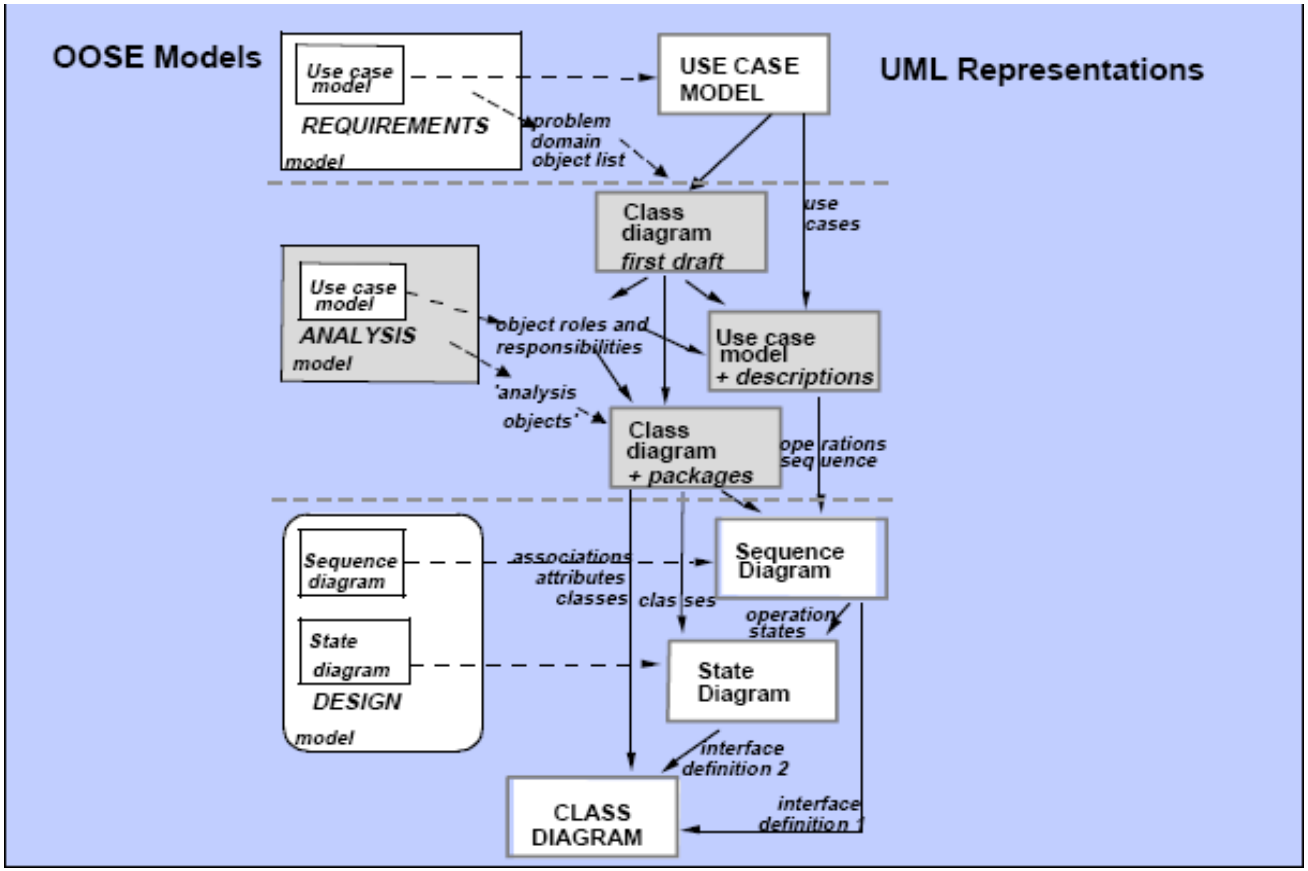
- STD merepresentasikan tingkah laku sistem dengan menggambarkan keadaan dan kejadian yang menyebabkan sistem mengubah keadaan
- Dalam STD, suatu aksi diambil sebagai akibat dari suatu kejadian khusus

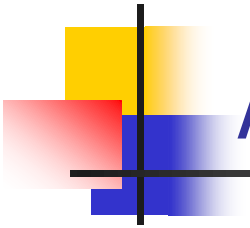
Pemodelan Tingkah Laku



Pendekatan OO (OOSE)

- UML





Analisis Kebutuhan (OOSE)

- Tujuan
  - Mendefinisikan masalah dan batasan/cakupan
  - Memberi gambaran kepada developer tentang cakupan masalah (tanpa harus detil)
- Perlu keterlibatan user
- Teknik pengumpulan informasi
  - Interview
  - Evaluasi dokumen
  - Sistem walkthrough

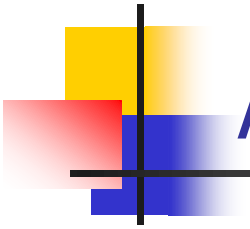
Analisis Kebutuhan (OOSE)

- Deskripsi → object, attribute, actor, function identification
  - buat vocabulary (i.e., data dictionary) dari problem domain
  - Identifikasi nouns → nouns adl kandidat kuat untuk attributes atau objects
  - Identifikasi adjectives → adjectives dpt menjadi attributes
  - Identifikasi verbs → verbs mengarah ke use cases
  - Identifikasi actors → orang/useryang terlibat dalam sistem

Analisis Kebutuhan (OOSE)

- Contoh deskripsi masalah : Aplikasi toko video

Create a video rental tracking system that allows any member with a credit card on file to rent any available video for a maximum of three days. Members can rent at most four videos at a time. When the member brings a video to the counter, the clerk scans or types the video identifier and the member's identification. The video is then rented. If the video is returned late, the member must pay Rp 2000 per day late fee. Members with outstanding late fees or overdue videos cannot rent movies.

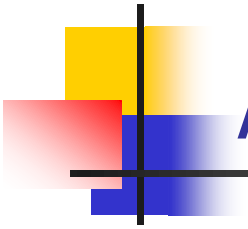


An Example: A Video Store Application

- Initial selection of application business objects and their attributes
  - Complex elements are most likely objects
  - Primitives (integer, string, etc.) are most likely predicates

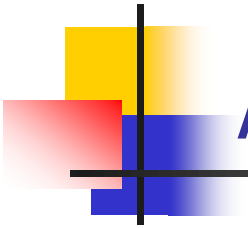
Member
- identifier
- fees outstanding
- late videos
- rented videos

Video
- identifier
- status (rented, late, returned)



An Example: A Video Store Application

- Identification of (preliminary) use cases
  - Rent a video (id, customer)
  - Return a rented video (id)
  - Add a new video to the inventory (id, title)
  - Create a new member (id, credit card)
  - Delete an existing video (id)
  - Delete an existing member (id)
  - Make a video late (id)
  - Pay overdue fees (id, amount)



An Example: A Video Store Application

- Finding outcomes of each use case
  - Find for logically invalid inputs
    - id does not exist
    - a rented video may already be rented
    - a returned video may have status indicating it was not rented before
  - Duplicating when creating new objects
    - created id may already exist
  - Business rules
    - a member with outstanding fines or videos may not rent
    - members have a max. number of videos that can be rented
    - videos are considered late after three days

An Example: A Video Store Application

Finalize the use cases

Rent a video (video id, member id)
{ OK, video not found, member not found, member has max. videos rented, member owes fines (amount) }
Return a rented video (video id)
{ OK, video not found, video not rented, video is overdue (fines owed) }
Add a new video to the inventory (video id, title)
{ OK, video id already exists }
Delete a video (video id)
{ OK, video not found, video is rented (due back) }
Create a new member (member id, credit card no.)
{ OK, duplicate member id, invalid credit card }
Delete an existing member (member id)
{ OK, member not found, member owes fines, member has outstanding rentals }
Pay fines owed by member (member id, amount)
{ OK, member not found, member owes no fines, member overpaid by (amount), member still owes fines (amount) }