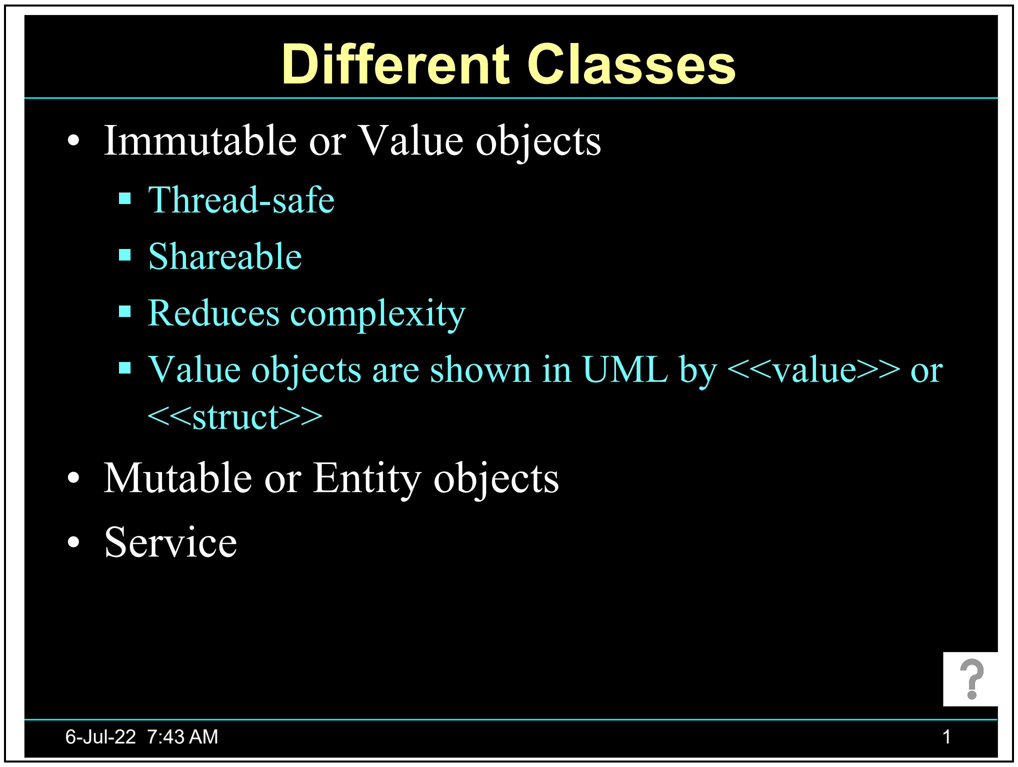


Different Classes

- Immutable or Value objects
  - Thread-safe
  - Shareable
  - Reduces complexity
  - Value objects are shown in UML by <<value>> or <<struct>>
- Mutable or Entity objects
- Service

6-Jul-22 7:43 AM 1

Service – Stateless.

Mutable. Keep state-space small, well-defined. Make clear when its legal to call which method.

Reference objects (Entities) and Value objects.

> Keep classes immutable unless there is a good reason to do otherwise.
>
> Enforcing contracts is a great way to reduce complexity. Simplifies development effort
>
> All instance variables should be “private final”.
>
> Value objects are not in any specific tier. They should be useable from all tiers.

================================

Immutable Example:

> Color, Line Style, Name, OrderNumber, ZipCode
>
> Company class has company address, company fax, company name, company Telephone, etc.
>
> Integers with limitations - Percentage(0 to 100%), Quantity (>=0)
>
> Arguments used in service methods
>
> Exceptions that are domain logical

JDK examples

Bad – Date, Calendar, Dimension

Good - TimerTask

Just as String does not belong to any tier.

For many applications makes sense to

Use immutable classes most of the time

Handle mutations only in a part of the application

This limits complexity to one part

Rest of the application uses immutable classes to reduce complexity and enhance understanding.

E.g. Historic information, read from the database, should be immutable.

Example: A credit card engine

It needs to read details about Customer, Card and Transaction.

In the scope of this engine, all these classes are immutable.

Getters should normally return immutable objects.

java.util.collections.unmodifiableList

System.Collections.Generic. IEnumerable<T> or List<T>.AsReadOnly

----

Entities have an identity. So two account entities with same balance are not the same. They are almost always persistent.

Values are usually a part of Entity.

DTO (Data Transfer Objects) are different from Value objects. DTO is a

-purpose: data transfer – technical construct

- bunch of data – not necessarily coherent

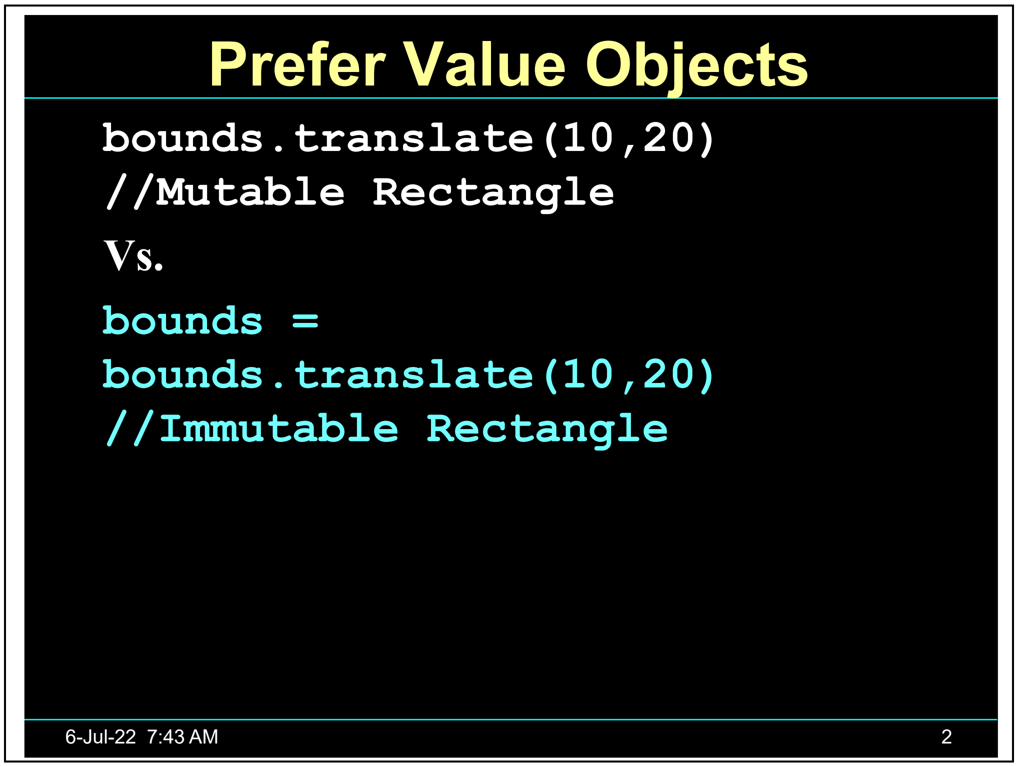
- no / little behavior

Value Object

-Purpose: domain representation

-High coherent data

-Rich on behavior



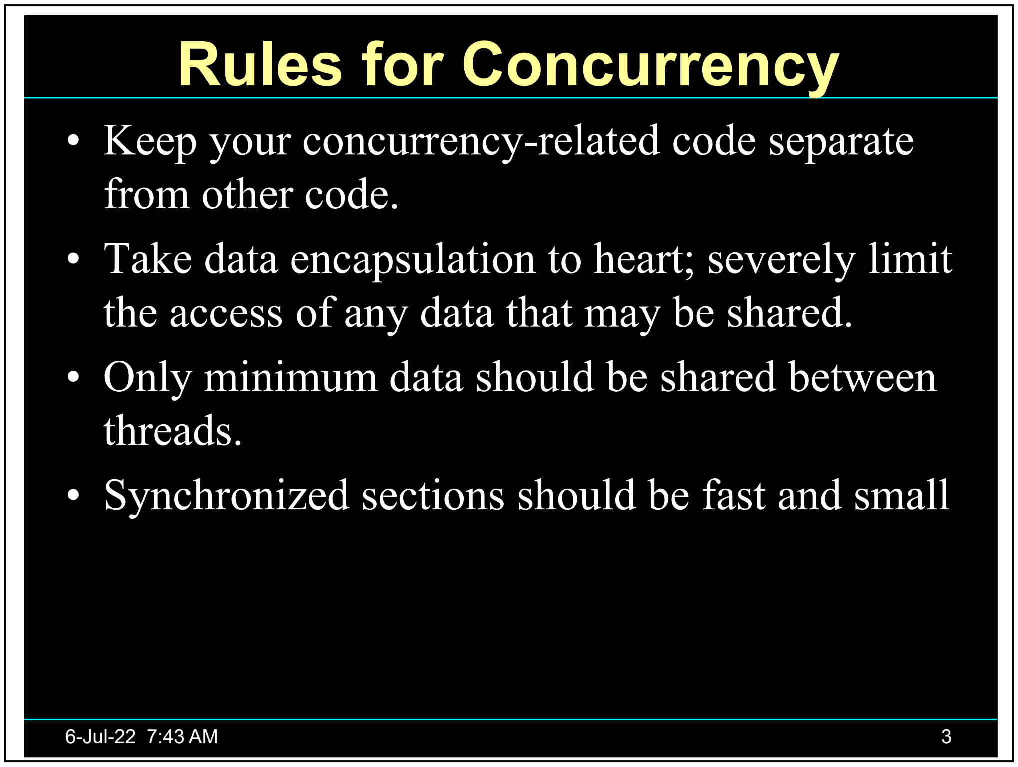
Prefer Value Objects

```
bounds.translate(10,20)
//Mutable Rectangle
```

Vs.

```
bounds =
bounds.translate(10,20)
//Immutable Rectangle
```

6-Jul-22 7:43 AM 2



Rules for Concurrency

- Keep your concurrency-related code separate from other code.
- Take data encapsulation to heart; severely limit the access of any data that may be shared.
- Only minimum data should be shared between threads.
- Synchronized sections should be fast and small

6-Jul-22 7:43 AM 3

Return copies of data or use immutable objects.