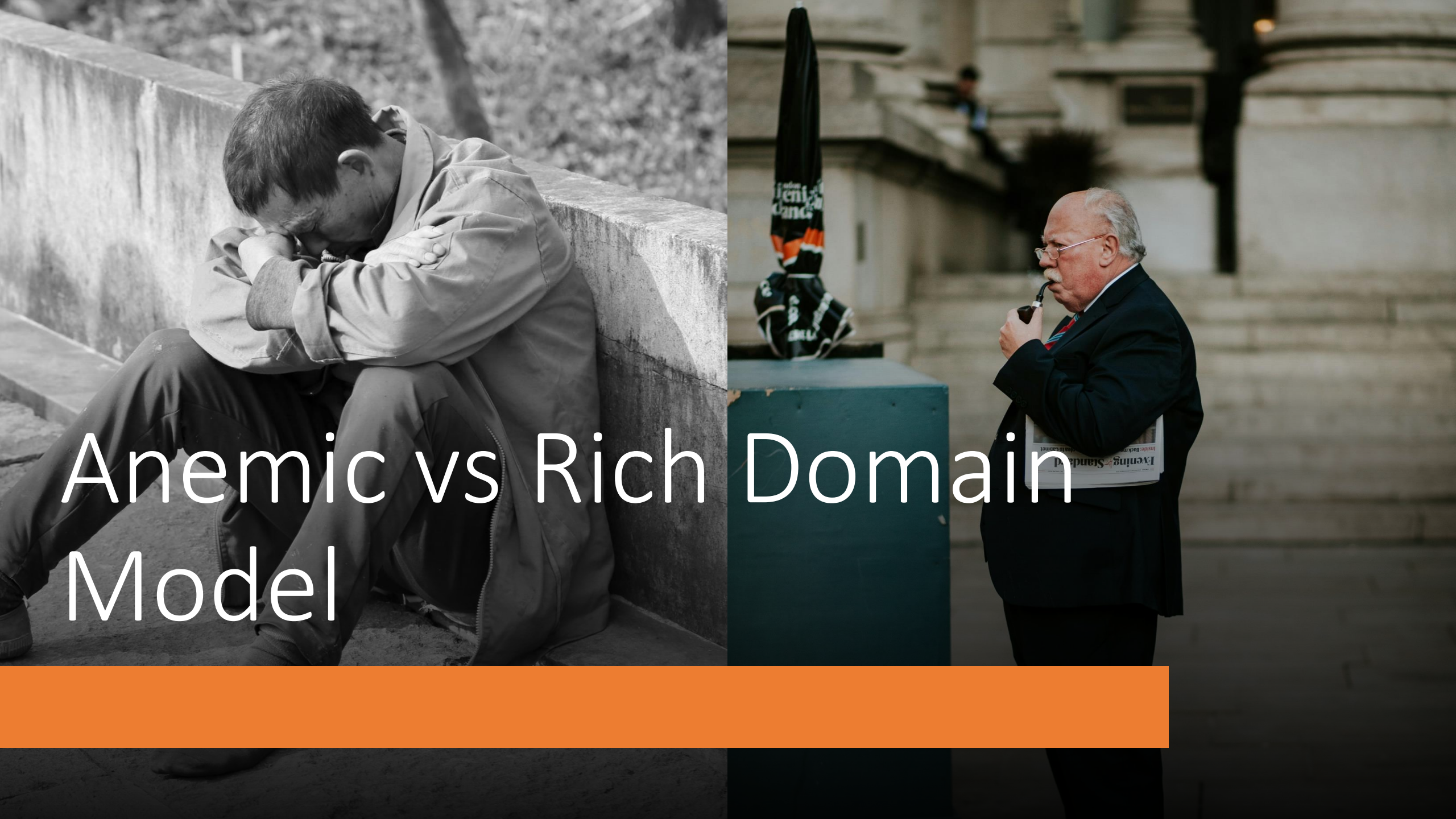
Java – Introducing Clean Code

By vebjorn.ohr@bouvet.no

Content

- Anemic Domain Model vs Rich Domain model
- Examples in Spring Boot Java project
- Discussion
- Conclusion

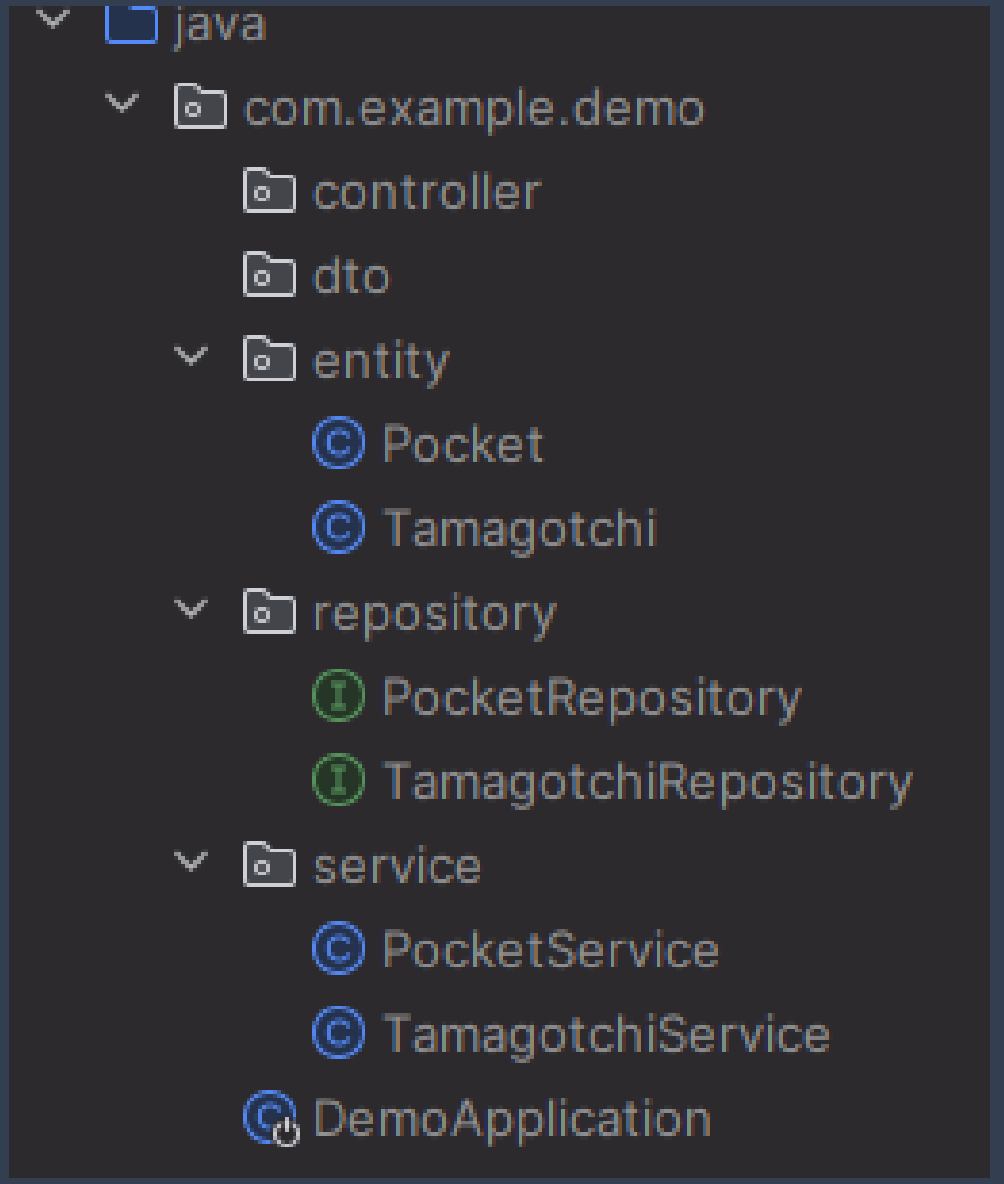


Anemic vs Rich Domain Model

Typical Spring Boot project

src/

- controller/
- dto/
- entity/
- repository/
- service/



Entity/domain classes

```java
@Entity
@NoArgsConstructor
@Getter
@Setter
public class Pocket {

    @Id
    private UUID id;

    private String name;

    @OneToMany(mappedBy = "pocket", cascade = PERSIST, orphanRemoval =
    private List<Tamagotchi> tamagotchis = new ArrayList<>();

}
```

```java
@Entity
@Getter
@Setter
@NoArgsConstructor
public class Tamagotchi {

    @Id
    private UUID id;

    private String name;

    @ManyToOne(fetch = LAZY)
    @JoinColumn(name = "pocket_id")
    private Pocket pocket;

}
```

Problems

```java
@Entity
@Getter
@Setter
@NoArgsConstructor
public class Tamagotchi {

    @Id
    private UUID id;

    private String name;

    @ManyToOne(fetch = LAZY)
    @JoinColumn(name = "pocket_id")
    private Pocket pocket;

}
```

```java
@Entity
@NoArgsConstructor
@Getter
@Setter
public class Pocket {

    @Id
    private UUID id;

    private String name;

    @OneToMany(mappedBy = "pocket", cascade = PERSIST, orphanRemoval = true)
    private List<Tamagotchi> tamagotchis = new ArrayList<>();

}
```

Service Classes

```java
@Service
@AllArgsConstructor
public class TamagotchiService {

    private final TamagotchiRepository tamagotchiRepository;
    private final PocketService pocketService;

    no usages
    @Transactional
    public void changeName(UUID id, String newName) throws Exception {
        if (!isValidName(newName)) {
            throw new Exception("Invalid name");
        }

        Optional<Tamagotchi> tamagotchi = tamagotchiRepository.findById(id);
        tamagotchi.orElseThrow();
        tamagotchi.get().setName(newName);
    }

    no usages
    @Transactional
    public Tamagotchi createTamagotchi(String name, UUID pocketId) {
        Optional<Pocket> pocket = pocketService.findById(pocketId);
        pocket.orElseThrow();
        Tamagotchi newTamagotchi = new Tamagotchi();
        newTamagotchi.setName(name);
        newTamagotchi.setId(UUID.randomUUID());
        newTamagotchi.setPocket(pocket.get());
        tamagotchiRepository.save(newTamagotchi);

        return newTamagotchi;
    }
```

Problems

```java
@Service
@AllArgsConstructor
public class TamagotchiService {

    private final TamagotchiRepository tamagotchiRepository;
    private final PocketService pocketService;

    no usages
    @Transactional
    public void changeName(UUID id, String newName) throws Exception {
        if (!isValidName(newName))
            throw new Exception("Invalid name");
        }

        Optional<Tamagotchi> tamagotchi = tamagotchiRepository.findById(id);
        tamagotchi.orElseThrow();
        tamagotchi.get().setName(newName);
    }

    no usages
    @Transactional
    public Tamagotchi createTamagotchi(String name, UUID pocketId) {
        Optional<Pocket> pocket = pocketService.findById(pocketId);
        pocket.orElseThrow();
        Tamagotchi newTamagotchi = new Tamagotchi();
        newTamagotchi.setName(name);
        newTamagotchi.setId(UUID.randomUUID());
        newTamagotchi.setPocket(pocket.get());
        tamagotchiRepository.save(newTamagotchi);

        return newTamagotchi;
    }
```

Sol

```
@Entit
@NoArg
@Sette
public
```

```
1 usag
publ


}
```

```java
public class Pocket {

    @Id
    @Getter
    private UUID id;

    private String name;

    @OneToMany(mappedBy = "pocket", cascade = PERSIST, orphanRemoval = true)
    private List<Tamagotchi> tamagotchis = new ArrayList<>();

    1 usage  new *
    public UUID createTamagotchi(TamagotchiCreateRequest request) {
        Tamagotchi tamagotchi = Tamagotchi.newTamagotchi(request.name(),  pocket: this);
        tamagotchis.add(tamagotchi);
        return tamagotchi.getId();
    }

    1 usage  new *
    public void updateTamagotchi(UUID tamagotchiId, TamagotchiUpdateRequest request) {
        Tamagotchi tamagotchi = tamagotchiById(tamagotchiId);
        tamagotchi.changeName(request.name());
    }

    1 usage  new *
    private Tamagotchi tamagotchiById(UUID tamagotchiId) {
        return tamagotchis
                .stream() Stream<Tamagotchi>
                .filter(t -> t.getId().equals(tamagotchiId))
                .findFirst() Optional<Tamagotchi>
                .orElseThrow(() -> new TamagotchiNotFoundException("Cannot find Tamagotchi by ID=" + tamagotchiId));
    }
```

```java
@Service
@AllArgsConstructor
public class PocketService {

    private final PocketRepository pocketRepository;


    no usages  Vebjørn Ohr
    public Optional<Pocket> findById(UUID pocketId) {
        return pocketRepository.findById(pocketId);
    }

    no usages  new *
    @Transactional
    public UUID createPocket(String name) {
        Pocket pocket = Pocket.newPocket(name);
        pocketRepository.save(pocket);
        return pocket.getId();
    }

    no usages  new *
    @Transactional
    public void updateTamagotchi(UUID tamagotchiId, TamagotchiUpdateRequest request) {
        UUID pocketId = pocketRepository.findByTamagotchiId(tamagotchiId);
        Pocket pocket = pocketRepository.findById(pocketId).orElseThrow();
        pocket.updateTamagotchi(tamagotchiId, request);
    }

    no usages  new *
    @Transactional
    public UUID createTamagotchi(UUID pocketId, TamagotchiCreateRequest request) {
        Pocket pocket = pocketRepository.findById(pocketId).orElseThrow();
        return pocket.createTamagotchi(request);
    }
```

Discussion

- Why is it so much used?
  - Old habits die hard?
  - Easier?
  - Unawareness?
- Are the frameworks and libraries to blame?

Conclusion

- Not so straight forward
- Both approaches have their uses
- Anemic model is less maintainable
- We lose benefits of object-oriented design
- Frameworks and old habits can hold us back

Sources

- https://dev.to/kirekov/rich-domain-model-with-hibernate-445k
- https://martinfowler.com/bliki/AnemicDomainModel.html
- This course

Questions?

Grazie Mille