CS 520

Theory and Practice of Software Engineering
Spring 2021

Best and worst programming practices

February 9, 2021

Reminder

Class website:
https://people.cs.umass.edu/~hconboy/class/2021Spring/CS520/

Schedule:

(subject to change; check regularly)

week	date	day	topic	homework	project
Week 1	Feb 2	Tu	Course introduction		
	Feb 4	Th	Software architecture and design		
Week 2	Feb 9	Tu	Best and worst programming practices		Project topic selection
	Feb 11	Th	Object oriented design principles		
Week 3	Feb 16	Tu	Object oriented design patterns		
	Feb 18	Th	User interfaces	Homework 1: Code review	
Week 4	Feb 23	Tu	Version control		
	Feb 25	Th	In-class exercise: Advanced uses of git		
	Mar 2	Tu	Software requirements		

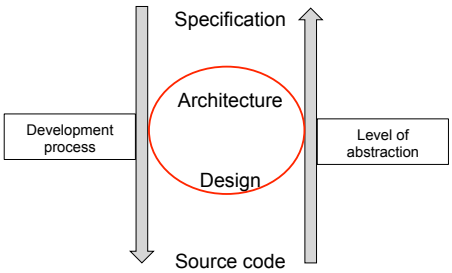
Reminder

Class website:
https://people.cs.umass.edu/~hconboy/class/2021Spring/CS520/

Instructor: Heather Conboy
Office hours: TBA and by appointment
Email: hconboy@cs.umass.edu

Graders:
Sayantan Bhowmik, Anjali Ramaprasad, Shashank Srigiri

Recap: software architecture vs. design

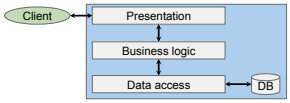


What's the difference?

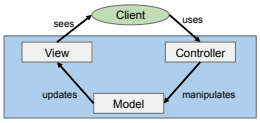
Recap: software architecture examples

- **Pipe and Filter**

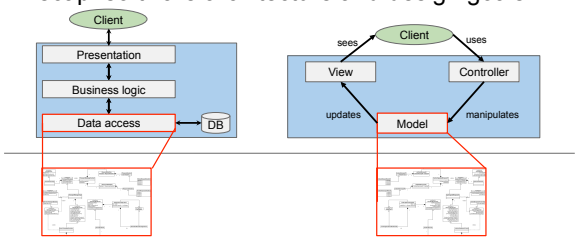
- **N-tier / Client-Server**



- **MVC (Model-View-Controller)**



Recap: software architecture and design goals



Architecture and design goals

- Lower complexity: separation of concerns, well defined interfaces
- Simplify communication
- Allow effort estimation and progress monitoring

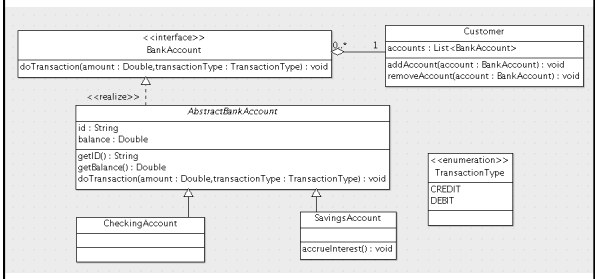
Example: Bank Account Requirements (in Natural Language)

1. Declare an interface for a *BankAccount* with a method for *doTransaction* that takes as input an amount (as a Double) and a transaction type (either CREDIT or DEBIT)
2. Declare an abstract class for an *AbstractBankAccount* with appropriate fields
3. Declare a concrete class for *CheckingAccount*
4. Declare a concrete class for *SavingsAccount* with a method *accrueInterest*()
5. Declare a concrete class for *Customer* who may have zero or more BankAccounts

Example: Bank Account Design (as a UML Class Diagram)

- Will use the ArgoUML editor: https://www.filehorse.com/download-argouml (for Mac or Windows)
- There are many other UML editors (and specifically UML class diagram editors)

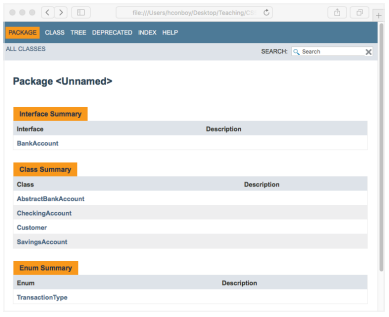
Example: Bank Account Design (as a UML Class Diagram)



Example: Bank Account Implementation (Written in Java)

```java
public enum TransactionType { CREDIT, DEBIT; }
public interface BankAccount {
  public void doTransaction(Double amount, TransactionType transactionType);
}
public abstract class AbstractBankAccount implements BankAccount {
  private String id;
  private Double balance;
  public String getID() { return null; }
  public Double getBalance() { return null; }
  public void doTransaction(Double amount, TransactionType transactionType) { }
}
public class CheckingAccount extends AbstractBankAccount { }
public class SavingsAccount extends AbstractBankAccount {
  public void accrueInterest() { }
}
public class Customer {
  private java.util.List<BankAccount> accounts;
  public void addAccount(BankAccount account) { }
  public void removeAccount(BankAccount account) { }
}
```

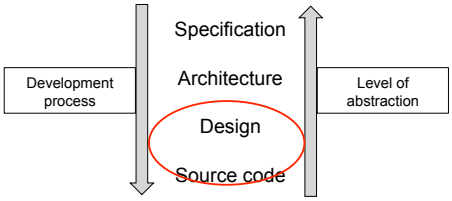
Example: Bank Account Documentation (Generated by javadoc)



https://www.oracle.com/java/technologies/javase/javadoc-tool.html

Today

- An in-class discussion on best and worst programming practices.



setup and goals

- 4-person teams
- 6 code snippets
- 4 rounds
  - **First round**
    - For each of 3 code snippets, decide whether it represents good or bad practice.
    - **Goal:** discuss and reach consensus on good or bad practice.
  - **Second round** (known solutions)
    - For each code snippet, try to understand why it is good or bad practice.
    - **Goal:** come up with one or more explanations or a counter argument.

  and then repeat with 3 more code snippets

Round 1: good or bad?

Snippet 1: good or bad?

```
public File[] getAllLogs(Directory dir) {
   if (dir == null || !dir.exists() || dir.isEmpty()) {
      return null;
   } else {
      int numLogs = … // determine number of log files
      File[] allLogs = new File[numLogs];
      for (int i=0; i<numLogs; ++i) {
         allLogs[i] = … // populate the array
      }
      return allLogs;
   }
}
```

Snippet 2: good or bad?

```
public void addStudent(Student student, String course) {
   if (course.equals("CS520")) {
      cs520Students.add(student);
   }
   allStudents.add(student)
}
```

Snippet 3: good or bad?

```
public enum PaymentType {DEBIT, CREDIT}

public void doTransaction(double amount, PaymentType payType) {
  switch (payType) {
    case DEBIT:
       … // process debit card
       break;
    case CREDIT:
       … // process credit card
       break;
    default:
       throw new IllegalArgumentException("Unexpected payment type");
   }
}
```

Solutions

- Snippet 1: bad
- Snippet 2: bad
- Snippet 3: good

Round 2: why is it good or bad?

Snippet 1: this is bad! why?

```
public File[] getAllLogs(Directory dir) {
   if (dir == null || !dir.exists() || dir.isEmpty()) {
      return null;
   } else {
      int numLogs = … // determine number of log files
      File[] allLogs = new File[numLogs];
      for (int i=0; i<numLogs; ++i) {
         allLogs[i] = … // populate the array
      }
      return allLogs;
   }
}
```

Snippet 1: this is bad! why?

```
public File[] getAllLogs(Directory dir) {
   if (dir == null || !dir.exists() || dir.isEmpty()) {
      return null;
   } else {
      int numLogs = … // determine number of log files
      File[] allLogs = new File[numLogs];
      for (int i=0; i<numLogs; ++i) {
         allLogs[i] = … // populate the array
      }
      return allLogs;
   }
}
```

```
File[] files = getAllLogs();
for (File f : files) {
…
}
```

Don't return null; return an empty array instead.

Snippet 2: short but bad! why?

```
public void addStudent(Student student, String course) {
   if (course.equals("CS520")) {
      cs520Students.add(student);
   }
   allStudents.add(student)
}
```

Snippet 2: short but bad! why?

```
public void addStudent(Student student, String course) {
   if (course.equals("CS520")) {
      cs520Students.add(student);
   }
   allStudents.add(student)
}
```

Defensive programming: write the literal first (or add an explicit assertion).

Snippet 3: this is good, but why?

```
public enum PaymentType {DEBIT, CREDIT}

public void doTransaction(double amount, PaymentType payType) {
  switch (payType) {
    case DEBIT:
       … // process debit card
       break;
    case CREDIT:
       … // process credit card
       break;
    default:
       throw new IllegalArgumentException("Unexpected payment type");
  }
}
```

Snippet 3: this is good, but why?

```
public enum PaymentType {DEBIT, CREDIT}

public void doTransaction(double amount, PaymentType payType){
  switch (payType) {
    case DEBIT:
       … // process debit card
       break;
    case CREDIT:
       … // process credit card
       break;
    default:
      throw new IllegalArgumentException("Unexpected payment ty…");
   }
}
```

Type safety using an enum; throws an exception for unexpected cases (e.g., future extensions of PaymentType).

Round 3: more snippets

Snippet 4: good or bad?

```
public BitSet(int size, boolean initialValue) {
  this.bitSet = new boolean[size];

  for (int i = 0; i < this.bitSet.length; i++) {
    this.bitSet[i] = initialValue;
  }
}
```

Snippet 5: good or bad?

```
public class ArrayList<E> {
   public E remove(int index) {
      …
   }
   public boolean remove(Object o) {
      …
   }
   …
}
```

Snippet 6: good or bad?

```
public class Point {
   private final int x;
   private final int y;

   public Point(int x, int y) {
      this.x = x;
      this.y = y;
   }
   public int getX() {
      return this.x;
   }
   public int getY() {
      return this.y;
   }
}
```

Solutions

- Snippet 1: bad
- Snippet 2: bad
- Snippet 3: good
- Snippet 4: bad
- Snippet 5: bad
- Snippet 6: good

Round 4: why is it good or bad?

Snippet 4: also bad! huh?

```
public BitSet(int size, boolean initialValue) {
  this.bitSet = new boolean[size];

  for (int i = 0; i < this.bitSet.length; i++) {
    this.bitSet[i] = initialValue;
  }
}
```

Snippet 4: also bad! huh?

```
public BitSet(int size, boolean initialValue) {
  this.bitSet = new boolean[size];

  for (int i = 0; i < this.bitSet.length; i++) {
    this.bitSet[i] = initialValue;
  }
}
```

Use run-time assertions to ensure
method pre- (and post-) conditions are satisfied

Snippet 5: Java API, but still bad! why?

```
public class ArrayList<E> {
   public E remove(int index) {
      …
   }
   public boolean remove(Object o) {
      …
   }
   …
}
```

Snippet 5: Java API, but still bad! why?

```
public class ArrayList<E> {
   public E remove(int index) {
      …
   }
   public boolean remove(Object o) {
      …
   }
   …
}
```

```
ArrayList<String> l = new ArrayList<>();
Integer index = new Integer(1);
l.remove(index);
```

Avoid method overloading, which is statically resolved.
Autoboxing/unboxing adds additional confusion.

Snippet 6: this is good, but why?

```
public class Point {
   private final int x;
   private final int y;

   public Point(int x, int y) {
      this.x = x;
      this.y = y;
   }
   public int getX() {
      return this.x;
   }
   public int getY() {
      return this.y;
   }
}
```

Snippet 6: this is good, but why?

```
public class Point {
   private final int x;
   private final int y;

   public Point(int x, int y) {
      this.x = x;
      this.y = y;
   }
   public int getX() {
      return this.x;
   }
   public int getY() {
      return this.y;
   }
}
```

Good encapsulation; immutable object.

Code reviewing

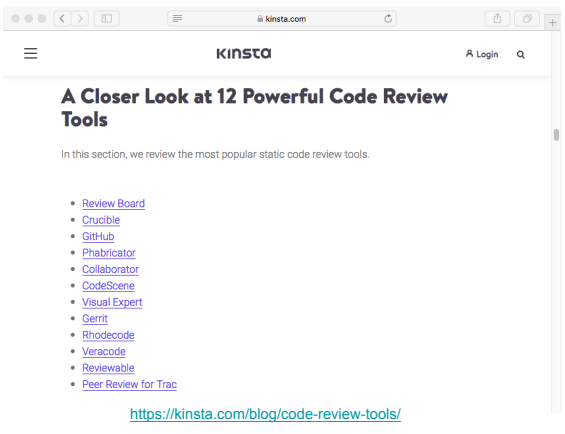
Code reviewing

Illustrated by the snippets:

- Decomposition into modules/packages, classes, and methods
- Encapsulation of methods and fields
- Type safety for methods (e.g., enum types instead of ints)
- Pre- and post-conditions for methods (e.g., defensive programming techniques, run-time assertions)

Additionally should consider:

- Naming conventions
- Documentation (e.g., internal comments, README files)
- Test suite



Final project description

- Each team of 4 will carry out **one** of the following projects:
  - MSR Mining Challenge
  - Replication Study (e.g., Automated Program Repair)
  - ML Development Toolkits (e.g., Weights & Biases)
  - EleNa: Elevation-based Navigation
- The key phases of the project are: topic selection, mid-point presentation, final presentation (and repository)
- More details available here: https://people.cs.umass.edu/~hconboy/class/2021Spring/CS520/finalProject.pdf