17-423/723 Recitation: Design for Testability **Feb 14, 2025**

Andrew IDs:

Consider a feature in an online shopping site that allows a customer to pay for an item in multiple installments over a period of time. For example, an item that costs $1000 can be paid in 4 installments of $250, with payments being made 1 month apart (e.g., Feb 14, March 15, April 15, May 15). The following class, **InstallmentGenerate**, is used to generate a series of installments and store them into a database.

```
public class InstallmentGenerator {

  // Handle to installment database
  private InstallmentDB db;

  public InstallmentGenerator(InstallmentRepository repository) {
     // Connect to the installment database at “dbAddress”
     this.db = InstallmentDB.connect(dbAddress);
  }

   public void generateInstallments(ShoppingCart cart, int numberOfInstallments) {
      // First installment begins today
      LocalDate nextInstallmentDueDate = LocalDate.now();
      // Calculate the amount per each installment
      double amountPerInstallment = cart.getValue() / numberOfInstallments;

      for(int i = 1; i <= numberOfInstallments; i++) {
         // Every installment is 30 days apart
         nextInstallmentDueDate = nextInstallmentDueDate.plusMonths(1);
         Installment newInstallment =
            new Installment(nextInstallmentDueDate, amountPerInstallment);
         // Store each installment in the database
         db.persist(newInstallment);
      }
   }
}
```

Q1. What are different test cases that should be considered to test the correctness of the installment generation feature?

Normal input:
cart.value() = 1000, numberOfInstallments = 4

Boundary cases:
cart.value() = 1000, numberOfInstallments = 0
cart.value() = 1000, numberOfInstallments = -5
cart.value() = 0, numberOfInstallments = 4
cart.value() = -50, numberOfInstallments = 4

numberOfInstallments = 15 (payment over a year)
numberOfInstallments = 4, nextInstallmentDueDate = Jan 30

- Test the corner case with February (e.g., Jan 30, Feb 28, Mar 30, April 30)

Q2. List one controllability challenge and one observability challenge in testing this component.

Controllability:

- It's difficult to test the component under a particular state of the database.
- It's difficult to test the component under a particular date for the first installment (e.g., nextInstallmentDueDate)

Observability:

- The method does not return any value, and it's difficult to test whether a correct list of installments is created for the given pair of input arguments.

Q3. What modifications would you make to the design of the component to improve its testability?

- Replace the reference to the database with an abstract Repository interface
- Inject repository into the class through the constructor; improve controllability
- Inject “today” into the class through the “generateInstallments” method; improve controllability
- Modify “generateInstallments” by having it return the list of installments; improve observability

```
public class InstallmentGenerator {

  // Handle to installment database
  private InstallmentRepository repository;

  public InstallmentGenerator(InstallmentRepository repository) {
    this.repository = repository
  }

  // requires: numberOfInstallments is between 1 and MAX_NUM_INSTALLMENTS
                cart != null and cart.value() >= 0
  // effects: finalpayment = cart.value() -
  //                              (numberOfInstallments - 1)*amount_per_installment
  public List<Installment> generateInstallments(ShoppingCart cart,
                           int numberOfInstallments, LocalDate today) {
    // First installment begins today
    LocalDate nextInstallmentDueDate = today;
    List<Installment> installments = new ArrayList();
    // Calculate the amount per each installment
    double amountPerInstallment = cart.getValue() / numberOfInstallments;

    for (int i = 1; i <= numberOfInstallments; i++) {
      // Every installment is 30 days apart
      nextInstallmentDueDate = nextInstallmentDueDate.plusMonths(1);
      Installment newInstallment =
        new Installment(nextInstallmentDueDate, amountPerInstallment);
      // Store each installment in the repository
      repository.persist(newInstallment);
      installments.add(newInstallment);
    }

    return installments;
  }
```

Q4. What are pre-conditions and post-conditions for the method **generateInstallments**?

- See the solution to **Q3.**