Activity 2 Refactoring Code handout

```
// Initial version
public class Main {
  private static String processGrades(Scanner scanner) {
    double t = 0;
    int c = 0;
    while (scanner.hasNextLine()) {
      scanner.next("\\s*\\w+\\s*");
      scanner.next("\\s*\\w+\\s*");
      String l = scanner.next("[ABCDFW][+-]?");
      if (scanner.hasNextLine()) {
        scanner.nextLine();
      }
      switch (l) {
      case "A":
        t += 4.00;
        break;
      case "A-":
        t += 3.67;
        break;
      case "B+":
        t += 3.33;
        break;
      case "B":
        t += 3.00;
        break;
      case "B-":
        t += 2.67;
        break;
      case "C+":
        t += 2.33;
        break;
      case "C":
        t += 2.00;
        break;
      case "C-":
        t += 1.67;
        break;
      case "D+":
        t += 1.33;
        break;
      case "D":
        t += 1.00;
        break;
      case "F":
        t += 0.00;
        break;
      case "W":
        break;
      default:
        throw new RuntimeException();
      }
      if (!l.equals("W")) {
        c += 1;
      }
    }
    double gpa = c == 0 ? 0 : t / c;
```

```
    return String.format("Courses: %d\nGPA: %.2f\n", c, gpa);
  }
}

// First refactoring
public class Main {
  private static String processGrades(Scanner scanner) {
    double total = 0;
    int courses = 0;
    while (scanner.hasNextLine()) {
      readPrefix(scanner);
      readCourseNo(scanner);
      String letter = readLetterGrade(scanner);
      readToEndOfLine(scanner);
      total += getGradeForLetter(letter);
      if (countsForCredit(letter)) {
        courses += 1;
      }
    }
    double gpa = computeGPA(courses, total);
    return formatSummary(courses, gpa);
  }

  private static void readPrefix(Scanner scanner) {
    scanner.next("\\s*\\w+\\s*");
  }

  private static void readCourseNo(Scanner scanner) {
    scanner.next("\\s*\\w+\\s*");
  }

  private static String readLetterGrade(Scanner scanner) {
    return scanner.next("[ABCDFW][+-]?");
  }

  private static void readToEndOfLine(Scanner scanner) {
    if (scanner.hasNextLine()) { scanner.nextLine(); }
  }

  private static double getGradeForLetter(String letterGrade) {
    switch (letterGrade) {
    case "A": return 4.00;
    case "A-": return 3.67;
    case "B+": return 3.33;
    case "B": return 3.00;
    case "B-": return 2.67;
    case "C+": return 2.33;
    case "C": return 2.00;
    case "C-": return 1.67;
    case "D+": return 1.33;
    case "D": return 1.00;
    case "F": return 0.00;
    case "W": return 0.00;
    default: throw new RuntimeException();
    }
  }

  private static boolean countsForCredit(String letter) {
```

```
    return !letter.equals("W");
  }

  private static double computeGPA(int courses, double total) {
    return courses == 0 ? 0 : total / courses;
  }

  private static String formatSummary(int courses, double gpa) {
    return String.format("Courses: %d\nGPA: %.2f\n", courses, gpa);
  }
}

// Extracting the Summary class
public class Main {
  private static String processGrades(Scanner scanner) {
    Summary summary = new Summary();
    while (scanner.hasNextLine()) {
      readPrefix(scanner);
      readCourseNo(scanner);
      String letter = readLetterGrade(scanner);
      readToEndOfLine(scanner);
      summary.addGrade(letter);
    }
    return summary.format();
  }

  private static void readPrefix(Scanner scanner) {
    scanner.next("\\s*\\w+\\s*");
  }
  private static void readCourseNo(Scanner scanner) {
    scanner.next("\\s*\\w+\\s*");
  }
  private static String readLetterGrade(Scanner scanner) {
    return scanner.next("[ABCDFW][+-]?");
  }
  private static void readToEndOfLine(Scanner scanner) {
    if (scanner.hasNextLine()) { scanner.nextLine(); }
  }

  static double getGradeForLetter(String letterGrade) {
    switch (letterGrade) {
    case "A": return 4.00;
    case "A-": return 3.67;
    case "B+": return 3.33;
    case "B": return 3.00;
    case "B-": return 2.67;
    case "C+": return 2.33;
    case "C": return 2.00;
    case "C-": return 1.67;
    case "D+": return 1.33;
    case "D": return 1.00;
    case "F": return 0.00;
    case "W": return 0.00;
    default: throw new RuntimeException();
    }
  }

  static boolean countsForCredit(String letter) {
    return !letter.equals("W");
```

```
  }
}

class Summary {
  private int courses = 0;
  private double total = 0.00;

  void addGrade(String letter) {
    this.total += Main.getGradeForLetter(letter);
    if (Main.countsForCredit(letter)) { this.courses += 1; }
  }

  String format() {
    return String.format("Courses: %d\nGPA: %.2f\n", courses, computeGPA());
  }

  private double computeGPA() {
    return courses == 0 ? 0 : total / courses;
  }
}

// Final version
public class Main {
  private static String processGrades(Scanner scanner) {
    Summary summary = new Summary();
    Processor processor = new Processor(scanner);
    while (processor.hasNext()) {
      summary.addGrade(processor.getNext());
    }
    return summary.format();
  }
}

class Processor {
  private final Scanner scanner;

  Processor(Scanner scanner) { this.scanner = scanner; }

  boolean hasNext() { return scanner.hasNextLine(); }

  Grade getNext() {
    readPrefix();
    readCourseNo();
    String letter = readLetter();
    readToEndOfLine();
    return new Grade(letter);
  }

  private void readPrefix() { scanner.next("\\s*\\w+\\s*"); }

  private void readCourseNo() { scanner.next("\\s*\\w+\\s*"); }

  private String readLetter() { return scanner.next("[ABCDFW][+-]?"); }

  private void readToEndOfLine() {
    if (scanner.hasNextLine()) { scanner.nextLine(); }
  }
}
```

```
class Summary {
  private int courses = 0;
  private double total = 0.00;

  void addGrade(Grade grade) {
    this.total += grade.getPoints();
    if (grade.countsForCredit()) { this.courses += 1; }
  }

  String format() {
    return String.format("Courses: %d\nGPA: %.2f\n", courses, computeGPA());
  }

  private double computeGPA() { return courses == 0 ? 0 : total / courses; }
}

class Grade {
  private final String letter;

  Grade(String letter) { this.letter = letter; }

  double getPoints() {
    switch (letter) {
    case "A": return 4.00; case "A–": return 3.67;
    case "B+": return 3.33; case "B": return 3.00; case "B–": return 2.67;
    case "C+": return 2.33; case "C": return 2.00; case "C–": return 1.67;
    case "D+": return 1.33; case "D": return 1.00;
    case "F": return 0.00; case "W": return 0.00;
    default: throw new RuntimeException();
    }
  }

  boolean countsForCredit() { return !letter.equals("W"); }
}
```