Assignment 02: Personal Greeter

CS 140 with Dr. Sam Schwartz

Due: Sunday, September 22 at 11:59pm via Canvas Upload

1 Purpose

The purpose of this assignment is to:

1. Deepen comfort with Java primitive/object types and named constants.
2. Practice robust console input using the `Scanner` class, including full-line input.
3. Demonstrate type casting (truncation vs. rounding) and string concatenation.
4. Produce clear, professional console output using both concatenation and `printf`-style formatting.

2 Tasks – High Level Overview

In this assignment, you will:

- Collect a small personal profile (name, age, height, preferences) from a user.
- Compute simple derived values (e.g., age in days; height conversions).
- Demonstrate multiple casts and formatted output styles.
- Present results twice: a quick greeting and a formatted "ID card" block.

3 Tasks – Detailed Requirements

3.1 Task 1: Setup

1. Create a new Java project called `Assignment02` in Eclipse.
2. Create a new class called `PersonalGreeter` with a `public static void main(String[] args)` method.
3. At the beginning of the file, include a multi-line comment containing:
   - Your name
   - A short description of what this program does.

3.2 Task 2: Constants & Variables

Within `PersonalGreeter.java`:

1. Declare at least **two named constants** using `final`, for example:
   - `DAYS_PER_YEAR = 365`

- `CM_PER_INCH = 2.54`

2. Declare at least **six variables of different types**, including:
   - A `String` for the user's full name (allow spaces).
   - A `String` for a preferred name or nickname.
   - An `int` for age in years.
   - A `double` for height in centimeters.
   - A `double` for a favorite decimal number.
   - A `boolean` for a yes/no preference (e.g., likes coffee).

3.3 Task 3: Prompt and Read Input

1. Create one `Scanner` to read from `System.in`.
2. Prompt for and read the user's **full name** using `nextLine()` (must support spaces).
3. Prompt for and read the **preferred name** using `nextLine()`.
4. Prompt for and read **age (int)**, **height in cm (double)**, **favorite number (double)** using the appropriate `Scanner` methods.
5. Prompt for and read a **yes/no** preference into a `boolean` variable. (You may read a short string such as "Y"/"N" and convert it to a boolean.)
6. If you mix `nextInt()`/`nextDouble()` with `nextLine()`, ensure you correctly handle the leftover newline (e.g., with an extra `nextLine()`). Add a short comment explaining why.

3.4 Task 4: Derived Values & Type Casting

1. Compute and store:
   - **Age in days** (approximate): age $\times$ `DAYS_PER_YEAR`.
   - **Height in inches**: cm $\div$ `CM_PER_INCH`.
2. Demonstrate **at least two distinct numeric casts**, for example:
   - Truncate height-in-inches to an `int`.
   - Cast a `double` to `int` to illustrate truncation *vs.* rounding.
3. Extract the user's **first initial** as a `char` from their preferred name.
4. Add a brief 2–3 line comment describing the difference between truncation by cast and rounding.

3.5 Task 5: Output — Two Presentations

1. **Quick Greeting (concatenation)**: Print one line greeting that uses string concatenation and includes:
   - Preferred name, age, favorite number (as entered), and first initial.
2. **ID Card (formatted block)**: Print a multi-line formatted block using `printf` (or `String.format`) that includes:
   - Full name and preferred name.
   - Age in years and approximate age in days.
   - Height in centimeters and in inches (show one with a fixed precision, e.g., 2 decimals).
   - The boolean preference (rendered as "Yes"/"No" or similar).

3. Use at least one field-width and one precision specification in your formatted output (e.g., `%.2f`).

3.6 What to Upload

You should upload **three** things to Canvas:

1. `PersonalGreeter.java`
2. A screenshot of the **Quick Greeting** output
3. A screenshot of the **ID Card** formatted output

4 Grading Criteria

Scale

Elements of nearly all assignments in this class will be broken down into a 0-1-2 scale.

- “2” means, “Nailed it!”
- “1” means, “Umm, kinda got it, but not really.”
- “0” means, “Uh-oh. Didn’t get it.”

The points are averaged out and multiplied by 100. The ceiling becomes the final score.
(I.e., if you get a 92.01%, we “round up” to 93%.)

Rubric

Validity

The student submitted a `PersonalGreeter.java` file to Canvas .... 2 1 0.
... which included a multi-line header comment with name and description .... 2 1 0.
... which declared at least two named constants using `final` .... 2 1 0.
... which declared at least six variables spanning multiple types .... 2 1 0.
... which used a single `Scanner` to collect all inputs .... 2 1 0.
... which correctly read full-line `String` input (names with spaces) using `nextLine()` .... 2 1 0.
... which correctly handled the newline when mixing `nextInt()/nextDouble()` with `nextLine()` (with an explanatory comment) .... 2 1 0.
... which converted a Y/N (or similar) input into a `boolean` .... 2 1 0.
... which computed age-in-days and height-in-inches from inputs and constants .... 2 1 0.
... which demonstrated at least two distinct numeric casts (and used each result) .... 2 1 0.
... which extracted a first initial as a `char` .... 2 1 0.
... which included a brief comment reflecting on truncation vs. rounding .... 2 1 0.
... which produced a Quick Greeting using concatenation .... 2 1 0.
... which produced an ID Card block using `printf/String.format` with field width and precision ... 2 1
0.

Screenshot

... which included a screenshot of the Quick Greeting output .... 2 1 0.
... which included a screenshot of the ID Card formatted output .... 2 1 0.

Readability

... and contained a sufficient number of comments explaining and documenting the code,
each of which was written in an appropriate register of professional English .... 2 1 0.

Fluency

... and the deliverable was executed in such a way that an experienced practitioner
would not find the deliverable “weird-in-a-bad-way” or unduly jarring .... 2 1 0.

Subtotal (of 36) ....

Total ($\lceil 100 \cdot \frac{\textbf{subtotal}}{36} \rceil$) ....