Java Problem Set Worksheet 1.I CMPE 261

Osman Emre Tutay

December 2025

Problem 1: Java Encapsulation

Design a Java class named `NuclearReactor` to demonstrate critical data protection.

- **Attributes:** `coreTemperature` (double) and `status` (String). Both must be **private**.
- **Methods:**
  - Public getter for `status`.
  - `adjustControlRods(double level)`: Decreases temperature based on insertion level.
  - `increasePower()`: Increases temperature.
  - **Logic:** If `coreTemperature` exceeds 2000.0, automatically set `status` to "MELTDOWN WARNING" and block further power increases.

Instructions:

- Attempt to increase power beyond the limit in the main method and verify the status change.
- Ensure `coreTemperature` cannot be modified directly.

Sample Output:

```
Power increasing... Temp: 1500.0
Power increasing... Temp: 2100.0
Reactor Status: MELTDOWN WARNING
Error: Safety protocols engaged. Cannot increase power.
```

Problem 2: Java Inheritance

Create a superclass `SocialMediaUser` and a subclass `Influencer`.

- **SocialMediaUser Class:** Attributes `username` (String) and `followerCount` (int). Include a constructor and a method `postContent()`.
- **Influencer Class:** Extends `SocialMediaUser`. Adds an attribute `sponsorshipDeals` (String list or array).
- **Constructor:** The `Influencer` constructor must initialize the inherited attributes and the new sponsorship list.

Instructions:

- Override `postContent()` in `Influencer` to include "#Sponsored by [Brand]" in the output.
- Instantiate an `Influencer` object and call the posting method.

Sample Output:

```
User: @TechGuru
Followers: 500000
Content Posted: Reviewing the new X-Phone! #Sponsored by TechCorp
```

Problem 3: Java Method Overloading

Create a class named `SmartKitchen` that handles automated cooking.

- Define a method `cook` that accepts a `String` (dish name) and prints "Preparing [dish]...".
- Overload `cook` to accept a `String` (dish) and `int` (minutes), printing "Slow cooking [dish] for [time] mins".
- Overload `cook` to accept a `String` (dish) and `boolean` (isSpicy), printing "Preparing [dish] (Spicy Mode: On/Off)".

Instructions:

- Call all three versions of `cook` from the main method with different inputs.

Sample Output:

```
Preparing Toast...
Slow cooking Stew for 120 mins
Preparing Tacos (Spicy Mode: On)
```

Problem 4: Java Method Overriding

Demonstrate runtime polymorphism using a base class `LoginMethod` and subclasses `FaceID` and `PINCode`.

- **LoginMethod Class:** Define a method `authenticate()` that prints "Standard authentication".
- **FaceID Class:** Override `authenticate()` to print "Scanning facial geometry... Access Granted."
- **PINCode Class:** Override `authenticate()` to print "Verifying 4-digit sequence... Access Granted."

Instructions:

- Create a `LoginMethod` reference variable.
- Assign a `FaceID` object to it and call `authenticate()`.
- Reassign a `PINCode` object to it and call `authenticate()`.

Sample Output:

```
Scanning facial geometry... Access Granted.
Verifying 4-digit sequence... Access Granted.
```

Problem 5: Java Abstraction

Create an abstract class `SpyGadget` and concrete subclasses `LaserWatch` and `SmokeBomb`.

- **SpyGadget Class:** Contains an abstract method `activate()`.
- **Subclasses:** Implement `activate()` to print the specific sound effect or action of the gadget.

Instructions:

- Create an array of `SpyGadget` objects containing one of each subclass.
- Use a loop to iterate through the array and activate all gadgets.

Sample Output:

```
LaserWatch: Cutting through steel door...
SmokeBomb: Creating visual cover...
```

Problem 6: Java Generics

Create a generic class `RecyclingBin<T>` to sort waste materials.

- **Attributes:** A private variable `wasteItem` of type `T`.
- **Methods:**
  - `throwAway(T item)`: Stores the item and prints "Discarded [item]".
  - `emptyBin()`: Returns the item and clears the bin.

Instructions:

- Create a `RecyclingBin` for `String` (e.g., "Plastic Bottle").
- Create a `RecyclingBin` for `Integer` (e.g., Electronic chip ID: 1010).
- Demonstrate throwing away items in both bins.

Sample Output:

```
RecyclingBin (String): Discarded Plastic Bottle
RecyclingBin (Integer): Discarded Chip ID 1010
```

Problem 7: Java Exception Handling

Write a Java program for a `RollerCoaster` entry system.

Instructions:

- Ask the user to input their height in centimeters.
- Use a `try-catch` block to handle:
  - `InputMismatchException`: If the user enters text instead of a number.
  - **Custom Logic**: Throw a generic `Exception` (or `IllegalArgumentException`) if the height is less than 120cm or greater than 250cm, with the message "Height requirement not met."

Sample Input 1:

```
Enter height (cm): five
```

Sample Output 1:

```
Error: Invalid input. Please enter a number.
```

Sample Input 2:

```
Enter height (cm): 110
```

Sample Output 2:

```
Error: Height requirement not met. Safety lock engaged.
```