Principles of Object-Oriented Programming in the Advertisement System

---

1. Encapsulation

- Encapsulation involves bundling data and methods that operate on that data within a class, providing controlled access via public methods.

Implementation:

Fields in the Advertisement class, like advertID, isPaid, and reviewStatus, are private and accessed through public getters and setters.

```
 public String getAdvertID() {

    return advertID;

public void setPaid(boolean isPaid) {

    this.isPaid = isPaid;

}
```

Ensures that data is modified only through controlled methods, protecting the integrity of the objects.

---

2. Abstraction

- Abstraction hides complex details and shows only the essential features of an object.

- **Implementation**:
  Each class focuses on a specific responsibility:
    - MarketingStaff handles advertisement creation and approval.
    - ReviewProcess evaluates ad suitability.
    - ArchiveManager deals with archiving outdated advertisements.

The review process abstracts the logic for assessing an advertisement's suitability:

```
public void reviewAdvertisementSuitability(Advertisement ad) {

   ReviewProcess reviewProcess = new ReviewProcess();

   reviewProcess.assessSuitability(ad);

}
```

3. Polymorphism

- Polymorphism allows objects to be treated as instances of their parent class, enabling the same interface to represent different behaviors.

- **Implementation**: Methods like approveAdvertisement() behave differently based on an advertisement's review and payment status:

```
if (ad.getReviewStatus().equals("Approved")) {
   if (ad.isPaid()) {
      System.out.println("Advertisement approved for processing: " + ad.getAdvertID());
   } else {
      ad.setReviewStatus("Not Paid");
   }
}
```

4. Modularity

- Modularity divides a program into distinct components that are self-contained and reusable.

- **Implementation**:
  Each class represents a module with a single responsibility:
  - PaymentProcessor handles payment operations.
  - ProcessingCenter processes approved ads for publication.
  - ArchiveManager manages the archival process.

Example of modular design:

```
ArchiveManager archiveManager = new ArchiveManager();

archiveManager.archiveUnusedAdvertisements(staff.advertisements);
```

5. Real-World Representation

- OOP maps real-world entities to objects in the program for intuitive design.
- **Implementation**:
    - An Advertisement object encapsulates all the details of an ad, such as content, appearanceDate, and reviewStatus, just like a real-world advertisement.
    - The MarketingStaff class mirrors a staff member's responsibilities in managing advertisements.
    - The ProcessingCenter simulates a system for handling publication tasks.

Conclusion

This system demonstrates the core principles of Object-Oriented Programming, ensuring clarity, maintainability, and scalability. By adhering to these principles, the design is well-structured and extensible, capable of handling future enhancements with ease.