Real-World Problem Statements Using Python Classes and Objects

Problem Set 1: Basics Only (Classes and Methods)

1. Create a class `Printer` that simulates a basic printer. Include methods to add a document to the queue and print all documents.

Sample Input:

```
printer = Printer()
printer.add_document("Report.pdf")
printer.add_document("Invoice.docx")
printer.print_all()
```

Sample Output:

```
Printing: Report.pdf
Printing: Invoice.docx
```

2. Create a class `Library` to manage a collection of books. Include methods to add a book and list all books.

Sample Input:

```
library = Library()
library.add_book("To Kill a Mockingbird")
library.add_book("1984")
library.list_books()
```

Sample Output:

```
Books in Library:
1. To Kill a Mockingbird
2. 1984
```

3. Create a class `Calculator` to perform basic operations: addition, subtraction, multiplication, and division.

Sample Input:

```
calc = Calculator()
print(calc.add(5, 3))
print(calc.multiply(4, 2))
```

Sample Output:

```
8
8
```

4. Create a class `ToDoList` with methods to add a task and display all tasks.

Sample Input:

```
todo = ToDoList()
todo.add_task("Buy groceries")
todo.add_task("Clean the house")
todo.show_tasks()
```

Sample Output:

```
To-Do List:
1. Buy groceries
2. Clean the house
```

5. Create a class `BankAccount` with methods to deposit, withdraw, and display the balance.

Sample Input:

```
account = BankAccount()
account.deposit(100)
account.withdraw(30)
account.show_balance()
```

Sample Output:

```
Balance: $70
```

6. Create a class `MovieTicket` with methods to set movie details and display them.

Sample Input:

```
ticket = MovieTicket()
ticket.set_details("Avatar", "5:00 PM", "$12")
ticket.show_details()
```

Sample Output:

```
Movie: Avatar
Time: 5:00 PM
Price: $12
```

7. Create a class `ShoppingCart` to add items and calculate the total price.

Sample Input:

```
cart = ShoppingCart()
cart.add_item("Apple", 3, 1.0)
cart.add_item("Banana", 2, 0.5)
cart.calculate_total()
```

Sample Output:

```
Total Price: $4.0
```

8. Create a class `Student` with methods to set details (name, age, grade) and display them.

Sample Input:

```
student = Student()
student.set_details("Alice", 15, "10th")
student.show_details()
```

Sample Output:

```
Name: Alice
Age: 15
Grade: 10th
```

9. Create a class `Vehicle` with methods to set and display vehicle details.

Sample Input:

```
vehicle = Vehicle()
vehicle.set_details("Car", "Toyota", "Camry")
vehicle.show_details()
```

Sample Output:

```
Type: Car
Make: Toyota
Model: Camry
```

10. Create a class `Clock` with methods to set and display the current time.

Sample Input:

```
clock = Clock()
clock.set_time("10:30 AM")
clock.show_time()
```

Sample Output:

```
Current Time: 10:30 AM
```

Problem Set 2: Using Constructors

11. Create a class `Book` with a constructor to initialize title, author, and price. Include a method to display the details.

Sample Input:

```
book = Book("The Alchemist", "Paulo Coelho", 10.99)
book.show_details()
```

Sample Output:

```
Title: The Alchemist
Author: Paulo Coelho
Price: $10.99
```

12. Create a class `Laptop` with a constructor to initialize brand, model, and price. Add a method to display details.

Sample Input:

```
laptop = Laptop("Dell", "XPS 15", 1200)
laptop.show_details()
```

Sample Output:

```
Brand: Dell
Model: XPS 15
Price: $1200
```

13. Create a class `Pet` with a constructor to initialize name, type, and age. Include a method to display the details.

Sample Input:

```
pet = Pet("Buddy", "Dog", 5)
pet.show_details()
```

Sample Output:

```
Name: Buddy
Type: Dog
Age: 5
```

14. Create a class `Employee` with a constructor to initialize name, position, and salary. Add a method to display the details.

Sample Input:

```
employee = Employee("John", "Manager", 50000)
employee.show_details()
```

Sample Output:

```
Name: John
Position: Manager
Salary: $50000
```

15. Create a class `Phone` with a constructor to initialize brand, model, and price. Include a method to display the details.

Sample Input:

```
phone = Phone("Apple", "iPhone 13", 999)
phone.show_details()
```

Sample Output:

```
Brand: Apple
Model: iPhone 13
Price: $999
```

16. Create a class `BankCustomer` with a constructor to initialize name and account balance. Add a method to display details.

Sample Input:

```
customer = BankCustomer("Alice", 2000)
customer.show_details()
```

Sample Output:

```
Name: Alice
Account Balance: $2000
```

17. Create a class `Game` with a constructor to initialize name and genre. Include a method to display the details.

Sample Input:

```
game = Game("Minecraft", "Sandbox")
game.show_details()
```

Sample Output:

```
Name: Minecraft
Genre: Sandbox
```

18. Create a class `HotelRoom` with a constructor to initialize room number, type, and price. Add a method to display the details.

Sample Input:

```
room = HotelRoom(101, "Deluxe", 150)
room.show_details()
```

Sample Output:

```
Room Number: 101
Type: Deluxe
Price: $150
```

19. Create a class `Restaurant` with a constructor to initialize name and cuisine type. Add a method to display details.

Sample Input:

```
restaurant = Restaurant("Olive Garden", "Italian")
restaurant.show_details()
```

Sample Output:

```
Name: Olive Garden
Cuisine Type: Italian
```

20. Create a class `Event` with a constructor to initialize event name, date, and location. Include a method to display details.

Sample Input:

```
event = Event("Conference", "2023-12-15", "New York")
event.show_details()
```

Sample Output:

```
Event Name: Conference
Date: 2023-12-15
Location: New York
```
