### **ASSIGNMENT 4: DESIGN OVERVIEW**

#### **Entities and Their Responsibilities**

**Program**

* **Attributes:**
  + name: The name of the program.
  + type: The type of the program.
  + startDate: The start date of the program.
  + endDate: The end date of the program.
* **Responsibilities:**
  + Encapsulates the details of a program, including its name, type, and duration.

**Venue**

* **Attributes:**
  + name: The name of the venue.
  + location: The location of the venue.
  + country: The country where the venue is located.
  + city: The city where the venue is located.
  + state: The state where the venue is located.
  + postalCode: The postal code of the venue’s location.
  + type: The type of the venue.
  + congregationName: The name of the congregation currently reserving the venue.
  + capacity: The seating capacity of the venue.
  + isReserved: A flag indicating if the venue is currently reserved.
  + program: A vector of Program objects associated with the venue (contains programs for which the venue has been reserved).
* **Responsibilities:**
  + Manages information about a venue, including its location, capacity, and associated programs.
  + Handles operations related to program management, such as reserving and freeing the venue.
* **Methods:**
  + bool supportsProgramType(const string &venueType, const string &programType): Checks if the venue supports a specific program type.
  + bool isAvailableForDates(const string &startDate, const string &endDate) const: Checks if the venue is available for the given date range.
  + void reserveForProgram(const string &startDate, const string &endDate, const string &programName): Reserves the venue for a specific program and date range.
  + bool isValidType(const string &type): Validates venue type.

**Congregation**

* **Attributes:**
  + name: The name of the congregation.
  + type: The type of congregation.
  + startDate: The start date of the congregation.
  + endDate: The end date of the congregation.
  + programs: A vector of Program objects associated with the congregation.
* **Responsibilities:**
  + Represents a gathering or event type with a specific duration.
  + Manages programs associated with the congregation, including adding and deleting programs.
* **Methods:**
  + bool isValidProgramType(const string &programType): Checks if the program type is valid for the congregation.
  + int addProgram(const string &programType, const string &programName, const string &pStartDate, const string &pEndDate): Adds a program to the congregation.
  + int deleteProgram(const string &programName): Deletes a program from the congregation.
  + void showPrograms() const: Displays all programs in the congregation.

#### **Inherited Classes**

**Responsibilities:**

* Represents specific types of congregations with additional validation for program types.

**Methods:**

* **Conference**
  + Conference(const string &name, const string &startDate, const string &endDate): Constructor.
  + bool isValidProgramType(const string &programType): Validates program types like "Workshop," "Main Conference," "Banquet."
* **Games**
  + Games(const string &name, const string &startDate, const string &endDate): Constructor.
  + bool isValidProgramType(const string &programType): Validates program types like "Ceremony," "Track and field," "Indoor games," "Water sports."
  + int addProgram(const string &programType, const string &programName, const string &pStartDate, const string &pEndDate): Adds a program with additional venue checks.
* **Convention**
  + Convention(const string &name, const string &startDate, const string &endDate): Constructor.
  + bool isValidProgramType(const string &programType): Validates program types like "Workshop," "Main Conference," "Banquet," "Food Court," "Exhibition."
* **Concert**
  + Concert(const string &name, const string &startDate, const string &endDate): Constructor.
  + bool isValidProgramType(const string &programType): Validates program types like "Pre-concert," "Main Concert."

#### **Managers and Their Responsibilities**

**VenueManager**

* **Responsibilities:**
  + Manages operations related to venues, including adding, deleting, and reserving venues.
  + Handles event management within venues.
* **Methods:**
  + int addVenue(const string &name, const string &location, const string &type, int capacity): Adds a new venue.
  + int deleteVenue(const string &name): Deletes a venue.
  + int reserveVenueForProgram(const string &venueName, const string &programName, const string &startDate, const string &endDate): Reserves a venue for a specific program.
  + void showVenues() const: Displays all venues.

**ProgramManager**

* **Responsibilities:**
  + Manages operations related to programs within congregations.
  + Displays programs associated with specific congregations.
* **Methods:**
  + int addProgram(const string &congregationName, const string &programType, const string &programName, const string &startDate, const string &endDate): Adds a program to a congregation.
  + int deleteProgram(const string &congregationName, const string &programName): Deletes a program from a congregation.
  + void showPrograms(const string &congregationName): Displays all programs in a specific congregation.

**CongregationManager**

* **Responsibilities:**
  + Manages the creation and deletion of congregations.
  + Displays the list of existing congregations.
* **Methods:**
  + int addCongregation(const string &name, const string &type, const string &startDate, const string &endDate): Adds a new congregation.
  + int deleteCongregation(const string &name): Deletes a congregation.
  + void showCongregations() const: Displays all congregations.

#### **Relationships**

* **VenueManager**: Manages multiple Venues.
* **ProgramManager**: Manages Programs across Congregations.
* **CongregationManager**: Manages multiple Congregations.
* **Venue**: Hosts multiple Programs.
* **Congregation**: Contains multiple Programs and can be associated with multiple Venues.

#### **Design Considerations**

* **Encapsulation**: Each class encapsulates its attributes and provides methods to manipulate them, ensuring internal state consistency.
* **Modularity**: Clear separation of responsibilities among classes (e.g., VenueManager for venues, CongregationManager for congregations).
* **Reusability**: The Program class is reused across venues and congregations, promoting code consistency and reuse.

