**Quiz 2**

You are working with the **TechFest Committee** to organize an annual **technology exhibition**. The system should manage **exhibitor registrations, sponsorships, and visitor tracking**.

**Requirements:**

1. **Implement a base class Event** to store the **event name, date, and venue**.
2. **Implement a derived class TechFest that:**

* Manages a **fixed number of exhibitor slots** (default **100**) and a **total sponsorship fund** (default **5,000,000**).
* Keeps track of **registered exhibitors, allocated sponsorship funds, and visitors (students & professionals)**.
* Explicitly create methods to add **students and professionals** to their respective visitor lists.

1. **Implement a final derived class TechFestManagement that:**

* Allows **exhibitor registration** (ensuring slots are not exceeded).
* Manages **sponsorship fund allocation** (ensuring funds are not overdrawn).

1. **Use constructor chaining** to initialize attributes.
2. **Ensure exhibitor slots and sponsorship funds remain constant once initialized.**

Solution:

#include <iostream>

#include <cstring>

using namespace std;

// Base class Event

class Event {

protected:

char name[50];

char date[20];

char venue[50];

public:

Event(const char\* eventName, const char\* eventDate, const char\* eventVenue) {

strcpy(name, eventName);

strcpy(date, eventDate);

strcpy(venue, eventVenue);

}

void displayEventInfo() {

cout << "Event: " << name << "\nDate: " << date << "\nVenue: " << venue << "\n";

}

};

// Derived class TechFest

class TechFest : public Event {

protected:

const int totalExhibitorSlots;

const int totalSponsorshipFund;

int exhibitorsRegistered;

int sponsorshipUsed;

int studentVisitors;

int professionalVisitors;

public:

TechFest(const char\* eventName, const char\* eventDate, const char\* eventVenue)

: Event(eventName, eventDate, eventVenue), totalExhibitorSlots(100), totalSponsorshipFund(5000000), exhibitorsRegistered(0), sponsorshipUsed(0), studentVisitors(0), professionalVisitors(0) {}

bool registerExhibitor() {

if (exhibitorsRegistered < totalExhibitorSlots) {

exhibitorsRegistered++;

return true;

}

return false;

}

bool allocateSponsorship(int amount) {

if (sponsorshipUsed + amount <= totalSponsorshipFund) {

sponsorshipUsed += amount;

return true;

}

return false;

}

void addStudentVisitor() { studentVisitors++; }

void addProfessionalVisitor() { professionalVisitors++; }

void displayTechFestInfo() {

displayEventInfo();

cout << "Total Exhibitor Slots: " << totalExhibitorSlots << "\nRegistered Exhibitors: " << exhibitorsRegistered << "\nRemaining Slots: " << (totalExhibitorSlots - exhibitorsRegistered) << "\n";

cout << "Total Sponsorship Fund: " << totalSponsorshipFund << "\nUsed Sponsorship: " << sponsorshipUsed << "\nRemaining Fund: " << (totalSponsorshipFund - sponsorshipUsed) << "\n";

cout << "Student Visitors: " << studentVisitors << "\nProfessional Visitors: " << professionalVisitors << "\n";

}

};

// Final derived class TechFestManagement

class TechFestManagement : public TechFest {

public:

TechFestManagement(const char\* eventName, const char\* eventDate, const char\* eventVenue)

: TechFest(eventName, eventDate, eventVenue) {}

void manageExhibitorRegistration() {

if (registerExhibitor()) {

cout << "Exhibitor registered successfully.\n";

} else {

cout << "No available exhibitor slots left.\n";

}

}

void manageSponsorship(int amount) {

if (allocateSponsorship(amount)) {

cout << "Sponsorship of " << amount << " allocated successfully.\n";

} else {

cout << "Insufficient sponsorship funds.\n";

}

}

};

int main() {

TechFestManagement event("TechExpo 2025", "May 15, 2025", "Convention Center");

event.displayTechFestInfo();

event.manageExhibitorRegistration();

event.manageSponsorship(1000000);

event.addStudentVisitor();

event.addProfessionalVisitor();

cout << "\nUpdated Event Information:\n";

event.displayTechFestInfo();

return 0;

}