**C++ Paradigm**

The solution effectively follows the object-oriented programming (OOP) paradigm by utilising several key language features of C++ :

**Encapsulation:**

The Player class encapsulates the player's balance, ensuring data integrity by using access control for private balance attributes and public methods.

**Abstraction:**

The SlotMachine class acts as an abstract base class, demonstrating abstraction by defining common behaviors and leaving the implementation details to its subclasses, NormalMachine and JackPotMachine.

**Inheritance:**

The inheritance relationship between the SlotMachine base class and its subclasses promotes code reuse and hierarchy, allowing subclasses to implement their variations of abstract virtual functions.

**Polymorphism:**

The chosenMachine pointer, which is of type SlotMachine but can point to either a JackPotMachine or a NormalMachine object, showcases polymorphism by allowing different objects to respond differently to the same method calls.

**Object Composition:**

The GameController class has a composition relationship with the Player object, as it contains an instance of the Player class as a member variable, demonstrating object composition.

The solution conforms to the OOP paradigm by utilising these language features effectively. It demonstrates a clear understanding of OOP principles by encapsulating data, abstracting common behaviors, promoting code reuse through inheritance, enabling polymorphic behavior, and composing objects to build larger systems. The implementation also emphasises modularity, supporting maintainability and extensibility by providing a straightforward and flexible framework for the slot machine game. Overall, the solution exhibits a practical application of OOP in C++, aligning with the stated paradigm.