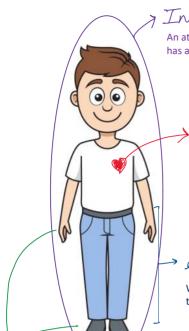
24 May 2024 01:02



## Inharitand: is about creating new things based on existing ones.

An athlete is a kind of human. An athlete can do everything a human can, but also has additional abilities like running fast.

## Encapsulation: is about keeping things together and protecting them.

Think of your heart inside your body. You can't directly touch your heart or see it working, but it keeps you alive by pumping blood. The heart is protected and hidden inside the body, and you interact with it through actions like exercise or rest.

Abstraction : is about simplifying complex systems by showing only the essential parts.

When you walk, you don't need to know how all the muscles and bones work together. You just know that your legs move, and that's enough.

Polymorphism: is about doing the same thing in different ways.

Both hands and feet can grab things, but they do it differently. Hands can pick up a pencil, and feet can grab a ball.

```
Encapsulation
#include <iostream>
class Heart {
public:
  void pump() {
    std::cout << "Pumping blood" << std::endl;
class Human {
private:
  Heart heart; // Encapsulated heart
public:
  void pumpBlood() {
    heart.pump(); // Only accessible through this method
};
int main() {
  Human person;
  person.pumpBlood(); // We interact with the heart through the
human
  return 0:
```

```
#include <iostream> Infution

class Human {
    public:
        void walk() {
            std::cout << "Walking" << std::endl;
        }
    };

class Athlete : public Human {
    public:
        void run() {
            std::cout << "Running fast" << std::endl;
        }
    };

int main() {
        Athlete athlete;
        athlete.walk(); // Inherited method
        athlete.run(); // New method
        return 0;
}
```

```
Polymorphism:
#include <iostream>
class Human {
public:
  virtual void move() {
    std::cout << "Moving" << std::endl;
};
class Athlete : public Human {
public:
  void move() override {
    std::cout << "Running fast" << std::endl;
};
class Dancer : public Human {
  void move() override {
    std::cout << "Dancing gracefully" << std::endl;
};
int main() {
  Human* people[] = { new Athlete(), new Dancer() };
  for (Human* person : people) {
    person->move(); // Different behaviors depending on the object
  // Cleanup
  for (Human* person : people) {
    delete person;
  return 0;
```

```
#include <iostream>

class Human {
    private:
        void moveLegs() {
            std::cout << "Legs are moving" << std::endl;
        }

public:
        void walk() {
            moveLegs(); // We abstract the complex details
        }

};

int main() {
        Human person;
        person.walk(); // We don't need to know how the legs move return 0;
}
```