BÀI 4 – ĐA HÌNH

Mục tiêu:

- *Học cách sử dụng các kỹ thuật up-casting và down-casting*
- *Nắm được nguyên lý về đa hình và biết cách ứng dụng vào các ví dụ cụ thể*

Thời gian: 90 phút

1. LÝ THUYẾT

Up-casting và down-casting

Chuyển đổi các kiểu dữ liệu đối tượng: nhìn nhận một đối tượng như kiểu của các lớp nằm trên cùng một cây phân cấp kế thừa

+ Up-casting: đi lên trên cây phân cấp kế thừa, nhìn nhận một đối tượng của lớp cơ sở như là đối tượng của một lớp dẫn xuất. Java tự động chuyển đổi
+ Down-casting: đi xuống dưới cây phân cấp kế thừa, nhìn nhận một đối tượng của lớp dẫn xuất như là đối tượng của một lớp cơ sở. Java không tự động chuyển đổi, phải ép kiểu.

Đa hình

Thực hiện cùng một công việc theo nhiều cách khác nhau

+ Đa hình phương thức: Gọi phương thức theo các cách khác nhau tùy theo tham số
+ Đa hình đối tượng: Gọi phương thức theo các cách khác nhau tùy theo đối tượng được gọi

2. BÀI TẬP

1. Bổ sung phương thức trừu tượng *inThongTin()* trong lớp Tài khoản ngân hàng (Account), các lớp con của lớp Tài khoản ngân hàng sẽ ghi đè phương thức này và thực hiện công việc in ra các thông tin liên quan đến tài khoản ngân hàng.

2. Trong lớp Ngân hàng kết tập các Tài khoản ngân hàng, xây dựng phương thức *inThongTinTaiKhoan()* thực hiện công việc in thông tin của tất cả các tài khoản trong ngân hàng.

3. Viết trong mã nguồn phương thức main tạo ra một đối tượng ngân hàng và một số đối tượng tài khoản ngân hàng khác nhau, sau đó thêm các tài khoản ngân hàng đó vào ngân hàng. Thực hiện *inThongTinTaiKhoan()* của đối tượng ngân hàng và xem kết quả.

3. ROBOCODE NÂNG CAO

Các robot thông thường sẽ thực hiện ngay khi lệnh được gọi. Để có thể kết hợp nhiều việc cùng lúc (ví dụ: quét và bắt) ta có thể viết các robot nâng cao bằng cách khai báo lớp kế thừa **AdvancedRobot**

Khi chúng ta viết mã nguồn cho các lớp robot nâng cao, thay vì việc gọi trực tiếp đến *fire, ahead, turnRight…* ta phải gọi đến các phương thức set* tương ứng. Khi chúng ta muốn robot thực hiện tất cả các yêu cầu đó cùng lúc, ta bắt buộc phải gọi đến phương thức *execute()*. Ví dụ

```
public class NewBot extends AdvancedRobot {

    public void run() {
        setAdjustRadarForGunTurn(true);
        while (true) {
```

```
            setTurnRadarRight(10000);
            execute();
        }
    }

    public void onScannedRobot(ScannedRobotEvent e) {
        setTurnRight(e.getBearing());
        setFire(3);
        setAhead(e.getDistance());
    }
}
```

Tập trung bắn vào một mục tiêu

Khi radar quét đến mục tiêu, ta sẽ muốn cố định hướng của radar theo đối phương. Để làm được việc này ta sẽ muốn gọi đến `setTurnRadarRight(e.getBearing());` Tuy nhiên e.getBearing() lại trả về góc của *xe tăng* với đối phương chứ không phải radar. Do đó, để cố định hướng của radar theo một mục tiêu ta phải xử lý được sự chênh lệch này

```
public void onScannedRobot(ScannedRobotEvent e) {
    // Lock on to our target (this time for sure)
    setTurnRadarRight(getHeading() - getRadarHeading() + e.getBearing());
    ...
```

Ví dụ

```
public class NarrowBeam extends AdvancedRobot {

public class NewBot extends AdvancedRobot {
    public void run() {
        setAdjustRadarForGunTurn(true);
        setTurnRadarRight(1000); // Quét một lượt lần đầu
        execute();
        while (true) {
                // Dịch chuyển radar một chút để gọi đến sự kiện onScannedRobot
            if (getRadarTurnRemaining() == 0) {
                setTurnRadarRight(1);
            }
            execute();
        }
    }
    public void onScannedRobot(ScannedRobotEvent e) {
        setTurnRight(e.getBearing()); // Quay về phía đối phương
        if (Math.abs(getTurnRemaining()) < 10) {
            // Tiến lại gần
            if (e.getDistance() > 200) {
                setAhead(e.getDistance() / 2);
            }
            // Nếu quá gần thì lùi lại
            if (e.getDistance() < 100) {
                setBack(e.getDistance() * 2);
            }
            setFire(3.0);
        }
        // Khóa radar
        setTurnRadarRight(getHeading() - getRadarHeading() + e.getBearing());
    }
    public void onRobotDeath(RobotDeathEvent e) {
        // Sau khi đã tiêu diệt được đối thủ, tiếp tục quét robot
        setTurnRadarRight(1000);
    }
}
```

4. BÀI TẬP VỀ NHÀ

Phát triển Robocode áp dụng kết hợp các kỹ thuật đã học