```java
/** A class representing a rover.
 * @since 1.0
 * @version 3.0
 * @author Ludovic Provost
 */
public class Rover {

    /** Right pedal for rover.
     * Creates a right pedal instance to
     * be used when right pedal is used.
     */
    public Pedal rightPedal = new Pedal( leftOrRight: "R");

    /** Left pedal for rover.
     * Creates a left pedal instance to
     * be used when left pedal is used.
     */
    public Pedal leftPedal = new Pedal( leftOrRight: "L");

    /** State of rover.
     */
    int state;
    //3   constant forward speed
    //2   accelerate forward
    //1   deaccelerate forward
    //0   rest
    //-1  deaccelerate backward
    //-2  accelerate backward
    //-3  constant backward speed

    /** Creates rover and sets state to rest.
     */
    public Rover(int state) {
        this.state = state;
    }

    /** Sets state of rover to new specified state.
     * @param newState new state of rover.
     */
    public void setState(int newState) {
        if (-3 > newState || newState > 3) {
            throw new IllegalArgumentException("New state must be between -3 and 3.");
```

```java
     */
    int state;
    //3   constant forward speed
    //2   accelerate forward
    //1   deaccelerate forward
    //0   rest
    //-1  deaccelerate backward
    //-2  accelerate backward
    //-3  constant backward speed

    /** Creates rover and sets state to rest.
     */
    public Rover(int state) {
        this.state = state;
    }

    /** Sets state of rover to new specified state.
     * @param newState new state of rover.
     */
    public void setState(int newState) {
        if (-3 > newState || newState > 3) {
            throw new IllegalArgumentException("New state must be between -3 and 3.");
        }
        state = newState;
    }

    /** Returns state of rover as a string.
     * @return state of rover.
     */
    public String printState() {
        return switch (state) {
            case (0) -> "rest";
            case (1) -> "deaccelerate forward";
            case (2) -> "accelerate forward";
            case (3) -> "constant forward speed";
            case (-1) -> "deaccelerate backward";
            case (-2) -> "accelerate backward";
            case (-3) -> "constant backward speed";
            default -> "no current states";
        };
    }
}
```

```java
/** A class representing pedals and their different behaviours.
 * @since 1.0
 * @version 3.0
 * @author Ludovic Provost
 */

public class Pedal {

    /** The placement (left or right) of pedal.
     * Makes the placement easily accessible.
     */
    String leftOrRight;

    /** Creates pedal with the specified placement.
     * @param leftOrRight the placement.
     */
    public Pedal(String leftOrRight) {
        if (leftOrRight == null || !(leftOrRight.equals("L") || leftOrRight.equals("R"))) {
            throw new IllegalArgumentException("The String must be R or L");
        }
        this.leftOrRight = leftOrRight;
    }

    /** The method returns the new state of the rover in the form of an int
     *  based on the given number of presses and the rover's current state.
     * @param numberOfPresses number of presses on pedal.
     * @param state current state of rover.
     * @return new state of rover.
     */
    public int press(int numberOfPresses, int state) {

        if (numberOfPresses > 2 || numberOfPresses == 0) {
            throw new IllegalArgumentException("Invalid number of presses.");
        }
        if (numberOfPresses == 1 && state == 0 && leftOrRight.equals("L")) {
            return 2;
        } else if (numberOfPresses == 2 && leftOrRight.equals("R")) {
            if (state == 2) {
                return 1;
            } else if (state == -2) {
                return -1;
            }
```

```
        } else if (numberOfPresses == 2 && leftOrRight.equals("R")) {
            if (state == 2) {
                return 1;
            } else if (state == -2) {
                return -1;
            }
        } else {
            throw new IllegalArgumentException("Your number of presses does nothing.");
        }
        System.out.print("Your number of presses does nothing. ");
        return state;
    }

    /** The method returns the new state of the rover in the form of an int based on the
     *  given number of seconds the pedal has been pressed and the rover's current state.
     * @param numberOfSeconds number of seconds the pedal is pressed.
     * @param state current state of rover.
     * @return new state of rover.
     */
    public int hold(int numberOfSeconds, int state) {
        if (numberOfSeconds != 5) {
            throw new IllegalArgumentException("The number of seconds must be 5.");
        }
        if (numberOfSeconds == 5) {
            if (state == 0 && leftOrRight.equals("L")) {
              return -2;
            } else if ((state == -1 || state == -2) && leftOrRight.equals("R")) {
                return -3;
            } else if ((state == 1 || state == 2) && leftOrRight.equals("R")) {
                return 3;
            } else {
                throw new IllegalArgumentException("Your number of seconds does nothing.");
            }
        }
        System.out.print("Your number of seconds does nothing. ");
        return state;
    }

}
```