

## The Nine Versions of the Mountain Climbing Problem

---

```
/*
 * File: MountainKare11.java
 * -----
 * The MountainKare11 subclass gets Karel to climb a simple
 * mountain, plant a flag, and descend to the ground. This
 * version works only for the simple world shown in the handout.
 */

import stanford.karel.*;

public class MountainKare11 extends Karel {
    public void run() {
        turnLeft ();
        move ();
        turnLeft ();
        turnLeft ();
        turnLeft ();
        move ();
        turnLeft ();
        move ();
        turnLeft ();
        turnLeft ();
        turnLeft ();
        move ();
        putBeeper ();
        move ();
        turnLeft ();
        turnLeft ();
        turnLeft ();
        move ();
        turnLeft ();
        move ();
        turnLeft ();
        turnLeft ();
        turnLeft ();
        move ();
        turnLeft ();
    }
}
```

```
/*
 * File: MountainKarel2.java
 * -----
 * The MountainKarel2 subclass gets Karel to climb a simple
 * mountain, plant a flag, and descend to the ground. This
 * version works only for the specific world shown in the
 * handout, but defines turnRight to simplify the code.
 */

import stanford.karel.*;

public class MountainKarel2 extends Karel {

    public void run() {
        turnLeft();
        move();
        turnRight();
        move();
        turnLeft();
        move();
        turnRight();
        move();
        putBeeper();
        move();
        turnRight();
        move();
        turnLeft();
        move();
        turnRight();
        move();
        turnLeft();
    }

    /* Turns Karel 90 degrees to the right */
    private void turnRight() {
        turnLeft();
        turnLeft();
        turnLeft();
    }
}
```

```
/*
 * File: MountainKarel3.java
 * -----
 * The MountainKarel3 subclass gets Karel to climb a simple
 * mountain, plant a flag, and descend to the ground. This
 * version works only for the specific world shown in the
 * handout, but includes moveToWall so that Karel can approach
 * the mountain and walk away from it on the other side. It
 * also extends SuperKarel rather than Karel so that turnRight
 * and turnAround are now primitive operations.
 */

import stanford.karel.*;

public class MountainKarel3 extends SuperKarel {

    public void run() {
        moveToWall();
        climbMountain();
        moveToWall();
    }

    /* Climbs the specific mountain shown in the handout */
    public void climbMountain() {
        turnLeft();
        move();
        turnRight();
        move();
        turnLeft();
        move();
        turnRight();
        move();
        putBeeper();
        move();
        turnRight();
        move();
        turnLeft();
        move();
        turnRight();
        move();
        turnLeft();
    }

    /* Moves Karel forward until it is blocked by a wall */
    private void moveToWall() {
        while (frontIsClear()) {
            move();
        }
    }
}
```

```
/*
 * File: MountainKarel4.java
 * -----
 * The MountainKarel4 subclass gets Karel to climb a simple
 * mountain, plant a flag, and descend to the ground. This
 * version works only for the simple mountain shown in the
 * handout, but defines stepUp and stepDown to simplify the
 * code.
 */

import stanford.karel.*;

public class MountainKarel4 extends SuperKarel {

    public void run() {
        moveToWall();
        climbMountain();
        moveToWall();
    }

    /* Climbs the specific mountain shown in the handout */
    private void climbMountain() {
        stepUp();
        stepUp();
        putBeeper();
        stepDown();
        stepDown();
    }

    /* Send Karel up the step ahead of it */
    private void stepUp() {
        turnLeft();
        move();
        turnRight();
        move();
    }

    /* Send Karel down the step ahead of it */
    private void stepDown() {
        move();
        turnRight();
        move();
        turnLeft();
    }

    /* Include moveToWall as in MountainKarel3 */
}

```

```
/*
 * File: MountainKarel5.java
 * -----
 * The MountainKarel5 subclass gets Karel to climb a simple
 * mountain, plant a flag, and descend to the ground. This
 * version can easily be tailored to work for a stair-step
 * mountain of any size, but doing so still requires changing
 * the program.
 */

import stanford.karel.*;

public class MountainKarel5 extends SuperKarel {

    public void run() {
        moveToWall();
        climbMountain();
        moveToWall();
    }

    /* Climbs a stair-step mountain whose size appears in the for statement */
    private void climbMountain() {
        for (int i = 0; i < 4; i++) {
            stepUp();
        }
        putBeeper();
        for (int i = 0; i < 4; i++) {
            stepDown();
        }
    }

    /* Include stepUp, stepDown, and moveToWall as in MountainKarel4 */
}

```

```
/*
 * File: MountainKarel6.java
 * -----
 * The MountainKarel6 subclass gets Karel to climb a simple
 * mountain, plant a flag, and descend to the ground. This
 * version attempts to generalize the program to solve a
 * stair-step mountain of any size, but fails to implement
 * the descent correctly.
 */

import stanford.karel.*;

public class MountainKarel6 extends SuperKarel {

    public void run() {
        moveToWall();
        climbMountain();
        moveToWall();
    }

    /* Buggy code to climb up and down a stair-step mountain of any size */
    private void climbMountain() {
        while (frontIsBlocked()) {
            stepUp();
        }
        putBeeper();
        while (frontIsClear()) {
            stepDown();
        }
    }

    /* Include stepUp, stepDown, and moveToWall as in MountainKarel5 */
}


```



```
/*
 * File: MountainKarel7.java
 * -----
 * The MountainKarel7 subclass gets Karel to climb a simple
 * mountain, plant a flag, and descend to the ground. This
 * version tries to generalize the program so that it can
 * climb a stair-step mountain of any size.
 */

import stanford.karel.*;

public class MountainKarel7 extends SuperKarel {

    public void run() {
        moveToWall();
        climbMountain();
        moveToWall();
    }

    /* Another buggy attempt to climb a stair-step mountain of any size */
    private void climbMountain() {
        while (frontIsBlocked()) {
            stepUp();
        }
        putBeeper();
        move();
        while (rightIsClear()) {
            dropDown();
        }
    }

    /* Drops down from the midair position just past a descending step */
    private void dropDown() {
        turnRight();
        move();
        turnLeft();
        move();
    }

    /* Include stepUp and moveToWall as in MountainKarel6 */
}

```



```
/*
 * File: MountainKarel8.java
 * -----
 * The MountainKarel8 subclass gets Karel to climb a simple
 * mountain, plant a flag, and descend to the ground. This
 * version generalizes the program so that it can climb a
 * stair-step mountain of any size.
 */

import stanford.karel.*;

public class MountainKarel8 extends SuperKarel {

    public void run() {
        moveToWall();
        climbMountain();
        moveToWall();
    }

    /* Climbs up and down a stair-step mountain of any size */
    private void climbMountain() {
        while (frontIsBlocked()) {
            stepUp();
        }
        putBeeper();
        move();
        while (rightIsClear()) {
            dropDown();
        }
    }

    /* Drops down from the midair position just past a descending step */
    private void dropDown() {
        turnRight();
        move();
        turnLeft();
        if (frontIsClear()) {
            move();
        }
    }

    /* Include stepUp and moveToWall as in MountainKarel7 */
}

```



```
/*
 * File: MountainKarel9.java
 * -----
 * The MountainKarel9 subclass solves the problem of climbing
 * a stair-step mountain of any size using the power of recursion,
 * which gives rise to a shorter program, but one that usually
 * takes more time to understand.
 */

import stanford.karel.*;

public class MountainKarel9 extends SuperKarel {

    public void run() {
        moveToWall();
        climbMountain();
        moveToWall();
    }

    /* Climbs a stair-step mountain recursively */
    private void climbMountain() {
        if (frontIsClear()) {
            putBeeper();
        } else {
            stepUp();
            climbMountain();
            stepDown();
        }
    }

    /* Include stepUp, stepDown, and moveToWall as in MountainKarel4 */
}

```