Introduction to Karel ++ Programming

A summary of: Karel++ A Gentle Introduction to the Art of Object-oriented Programming
– By Bergin, Stehlik, Roberts & Pattis

Presented by:
Jim Polzin
Karel++ Basics: Introduction

World:

Robots:

Karel++ Basics:

Initial Situation:

Dangers:

Karel++ Overview and Object-Oriented Programming:

The Basic Robot Class - UR_Robot:

Primitive Instructions:

Comments:

Basic Program Structure:

The “Other” Basic Robot Class - Robot:

Boolean Tests:
KAREL++ BASICS

World:

- All robots function on a grid of streets and avenues
- There is a permanent wall boundary on the southern and western edges of the grid

Robots:

- Can only travel on the streets and avenues
- Can only move one block in the direction they are facing
- Are always located at the intersection of a street and avenue
- Are only able to turn $90^\circ$ to the left
- Can carry any number of beepers in their beeper bag
- Can pick up one beeper at a time or put down one beeper at a time
KAREL++ BASICS

Initial Situation:
- The world may contain additional wall segments, depending on the problem
  - Wall segments are always between adjacent intersections
- The world may contain beepers at any intersection, depending on the problem

Dangers:
- Robots cease functioning if they hit a wall
  - A robot remains at the last intersection it occupied when it hits a wall
- Robots cease functioning if they try to pick up a beeper that isn’t there
- Robots cease functioning if they try to put down a beeper that they don’t have
KAREL++ OVERVIEW AND OBJECT-ORIENTED PROGRAMMING

Object-oriented programming

• allows the programmer to more closely model the problem space
  - a programmer can talk about objects and how they work instead of always having to translate the problem into basic data types and functions
classes - general categories of things/objects are defined
objects - particular instances of a class or actual examples of a class definition

• methods & attributes
  - methods are the things an object can do (functions)
  - attributes are the characteristics of an object (data)

• encapsulation
  - the bundling of an object’s methods and attributes so that they are, or at least appear to be, all part of the object

• data/information hiding
  - not all the methods and attributes are necessarily available to the outside world
  - the internal structure of the object is masked behind the representation the interface presents to the outside world

• inheritance
  - the expression of relationships between classes
  - establishes an “is-a” relationship between an existing class and a new class definition

• code reuse/maintenance
  - many features of object-oriented languages lend themselves to minimizing programming effort and preventing programmers from damaging existing code

Karel++

• maintains basic C/C++ syntax
• supports object-oriented programming; but only with robot objects
THE BASIC ROBOT CLASS - UR_ROBOT

• The five primitive instructions are implemented in a fundamental type of robot.
• This fundamental type is referred to as a “class” of robot and is named “ur-Robot.”
• Any additional instructions or methods must be implemented in a new class of robot that “extends” the capabilities of the ur_Robot class using inheritance.

PRIMITIVE INSTRUCTIONS

• The five primitive instructions are: move, turnLeft, pickBeeper, putBeeper, turnOff
• All the primitive instructions are implemented as methods and are invoked as follows:

Ex:

```java
someRobot.move();
someRobot.turnLeft();
someRobot.pickBeeper();
someRobot.putBeeper();
someRobot.turnOff();
```
COMMENTS

- commenting to end-of-line using // is allowed
- block comments using /* and */ are also allowed

BASIC PROGRAM STRUCTURE

- A Karel++ program executes from a main module labeled task
- The task module, methods, class definitions, and blocks of code are all enclosed within braces, similar to C
- Individual statements and class definitions are terminated by semicolons
  
  NOTE: Method definitions are not terminated by semicolons, again like C

Ex:

```karel++
/* FILE: ex1.kpp */
/* This program has an ur_Robot place beepers on the corner positions of a square 2 blocks on a side. */
/* The main module of a Karel++ program is "task" */

task
{
    ur_Robot Karel(1, 2, East, 4); // This creates an ur_Robot named Karel
       // ... at 1 st. 2 ave. facing East with
       // ... four beepers in its bag.
    Karel.move(); // Sends a move() message to Karel
    Karel.move();
    Karel.putBeeper(); // Sends a putBeeper() message to Karel
    Karel.move();
    Karel.move();
    Karel.putBeeper();
    Karel.move();
    Karel.move();
    Karel.move();
    Karel.putBeeper();
    Karel.move();
    Karel.move();
    Karel.move();
    Karel.move();
    Karel.putBeeper();
    Karel.move();
    Karel.move();
    Karel.putBeeper();
    Karel.move();
    Karel.move();
    Karel.move();
    Karel.move();
    Karel.move();
    Karel.putBeeper();
    Karel.move(); // Sends a turnOff() message to Karel
}
/* The task definition is enclosed in a set of braces */
```
THE “OTHER” BASIC ROBOT CLASS - ROBOT

- This class of robot is named “Robot” – which we’ll refer to as “Big R – Robot”
- It is derived from the ur_Robot class and therefore has all the functionality of an ur_Robot
- In addition to the five primitive instructions inherited from ur_Robot, this class has a set of eight Boolean methods: frontIsClear(), nextToABeeper(), anyBeepersInBeeperBag(), facingNorth(), facingSouth(), facingEast(), facingWest(), and nextToARobot()
- These methods allow a programmer to test for certain conditions in their code, so that a program can do different things depending on the situation it is being executed in.

BOOLEAN TESTS

- The eight Boolean methods are passed no parameters and return either true or false.
- These tests are used in conditional if statements and while loops.

Ex:

```java
if (someRobot.frontIsClear()){  
someRobot.move();  
}

if (!someRobot.facingSouth()){  
someRobot.turnLeft();  
}

while (someRobot.nextToABeeper()){  
someRobot.pickBeeper();  
}

while (someRobot.anyBeepersInBeeperBag()){  
someRobot.putBeeper();  
}
```