1001ICT Programming 1 2013-2
Project

School of Information and Communication Technology
Griffith University

October 3, 2013

Due	Friday, teaching week 14, 5.00 pm
Goals	In the project you get to work on some programming problems that are larger than can be dealt with in a 2 hour laboratory class, and with more attention paid to the presentation of the source code.
Marks	35
Robots	Arm-NXT, Arm-NXT Track

1 Background

The theme for this project is a factory. The factory uses robots on the production line. A common application of industrial robots is to assist in moving parts and products from one place to another. Such robots could be mobile for longer distances or stationary for short distances. The robot for this project is of the latter kind. The management of the factory will use programs for factory operations monitoring, planning, and promoting factory operations and products.

2 Exercises

All of the following programming exercises are to be implemented in MaSH, and should all have at least a `main()` method. That means that programs that don't use methods at all can not get full marks.

Start work early. That way, if you get stuck, there are opportunities to ask questions, and you can make the best use of the time in which you have access to robots.

2.1 Program 1 (nxt, 6 marks)

Write a program that enables the Arm-NXT robot to be controlled manually using the NXT's front panel buttons (LEFT, RIGHT, ENTER for up, and ESCAPE for down) and the touch sensor (to alternately open and close the claw), as demonstrated in this video.

Hints:

- Use button handlers to respond to both pressing and releasing the NXT's front panel buttons.
- Use the touch sensor on the claw to stop the motor when the arm is lowered as far as it can go. The touch sensor on the gripper is used to detect the ground.
- The claw should alternate between opening and closing when the touch sensor on the NXT is pressed.
- Use of threads may make the coding simpler.

2.2 Program 2 (nxt, 6 marks)

Write the program that controls the Arm-NXT automatically, as shown in this video.

The robot must be started with the arm rotated somewhere between the two black marks on the track. The black wheels can be used to manually adjust the position of the arm. The gearing in the base and lift mechanisms make it impossible to just force it to move.

When started, it carries out these actions:

- The gripper claw is closed, then opened fully.
- The arm is lowered to the ground.
- The arm is rotated to the left black mark on the track at the pick-up zone.
- The arm is rotated to the right black mark at the drop zone. During this movement a rotation sensor is used to measure how much rotation is required to get from the pick-up zone to the drop zone, so that the same rotation may be performed later with the arm raised (making the light sensor on the bottom of the gripper useless).
- The arm is raised.
- The arm is rotated back to the *home position*, that is two thirds of the way back to the pick-up zone.
- Wait for a ball to be placed for pick-up.
- Pick up a ball from the pick-up zone, drop it over the drop zone, and return to the home position.
- Repeat the last two actions until the NXT is turned off.

Hints:

- Use time to control how long the motor runs when closing the claw. If it was already closed, a wait for rotation would never end. Use rotation to open a known amount.
- Use low motor powers for opening and closing the claw, but high powers for turning and lifting.
- This problem is much easier to write if you make good use of procedures. Write a procedure for every basic action the arm performs.
- The robot does not have to do more than one thing at a time. Threads are not necessary.

2.3 Program 3 (console, 6 marks)

In a given month, a factory will receive a lot of small orders for its various products. The management needs a program that adds up all the individual orders and produces a list of the totals for each product. For example the factory that makes the robots for this course (I wish) has the orders for the month in the file `Robots.txt`, available for download here.

```
Arm 2
Doorbell 3
Arm 1
Debris 3
Arm 1
Doorbell 1
Debris 1
Doorbell 3
Arm 1
Doorbell 2
Debris 7
Doorbell 3
Arm 1
Doorbell 2
Debris 3
Arm 3
Doorbell 3
Arm 1
Arm 1
Debris 1
Doorbell 1
Arm 2
Doorbell 1
```

```
Arm 1
Debris 1
Doorbell 1
Debris 3
Arm 1
Debris 1
```

Your program should work as follows. It should read the input file via standard input using input redirection (`<`) and print the total numbers of distinct products. Note that in this example `$` is the command line prompt which varies from system to system and does not have to be typed.

```
$ java TotalProducts < Robots.txt
Arm 15
Doorbell 20
Debris 20
$
```

You may assume that there are never more than 10 distinct product kinds in the input file. You can not assume what the products are. This example runs the same program with a different input file, `Products.txt`, available here.

```
$ java TotalProducts < Products.txt
Bandage 210
Armour 58
Ring 137
Sword 50
Helmet 50
Elixir 255
Boots 80
Bag 50
Scroll 127
Potion 98
$
```

2.4 Program 4 (graphics, 6 marks)

The management needs a program to represent the product order totals graphically. The input for this program is the same as the output of program 3, that is the names and totals for each product. This is an example run of both programs where the output of the first program is *piped* into the input of the second, using `|`. The output is displayed in figure 1.

```
$ java TotalProducts < Products.txt | java Column
$
```

You may assume that there are never more than 10 distinct product kinds in the input file. You can not assume what the products are. You may also assume that the total for each product never exceeds 1000.

The basic requirements for this program are to produce the basic column chart shown, with axes, fixed scales (up to 1000 of up to 10 distinct products), tick marks and labels on both axes. The program must read its input from standard input. Meeting the basic requirements is all that is required for full marks. However, the chart could be improved in many ways if you wish: sort the product names; make the scales variable; add colour; add a 3-dimensional effect.

2.5 Program 5 (files, 6 marks)

The factory management needs to know what parts are required to fill the product orders. For each of the products there is a parts list stored in a text file, for example `Arm.txt`:

```
20 Beam
10 Brick
25 Plate
10 Cog
3 Motor
3 Sensor
1 NXT
```

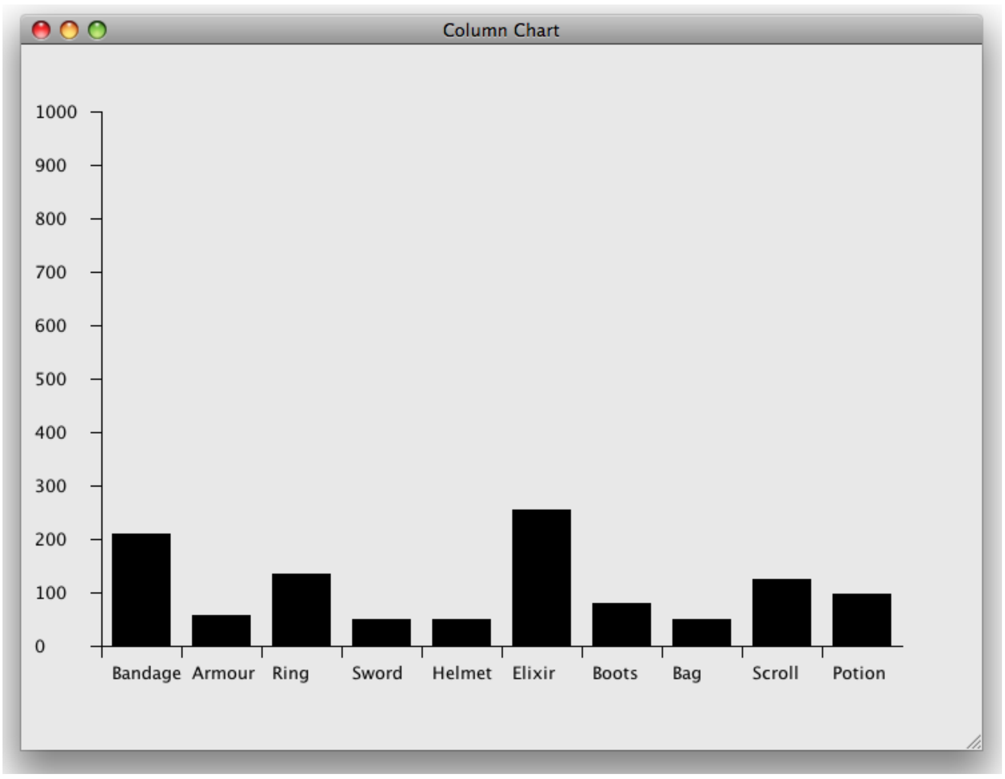

Figure 1: Totals for each product.

Write a program that reads the totals for each distinct product like that produced by program 3 and prints the totals required for each part. For example:

```
$ java TotalProducts < Robots.txt | java TotalParts
Beam 300
Brick 950
Plate 375
Cog 150
Motor 45
Sensor 65
NXT 35
$
```

Since the output format is the same as for program 3, program 4 can also plot these results, as displayed in figure 2
You may assume that all of the products can be made from combinations of only 10 distinct parts.

2.6 Program 6 (graphics, 5 marks)

The marketing department wants an animated display for their reception area. The display is to suggest the factory's products rolling along the production line. They have supplied 8 image files of products. Write a program that loads these 8 images and continuously scrolls images across a window, with the images selected at random.

Additionally, the display should be interactive. Somehow, via mouse or keyboard, a visitor should be able to get the name of a robot to pop up temporarily. Put a comment at the top of your program to tell the markers how to interact with your program.

The images are available for download here. Unzip the file and place the `.jpg` files in the same folder as your `.mash` program file. You don't need to submit these files with your project. We will add them for you. If you want, you may use additional images or sounds. If you do so, send them by email to `a.rock@griffith.edu.au`. Do not put them in an archive, *especially not a* `.rar` *file.*

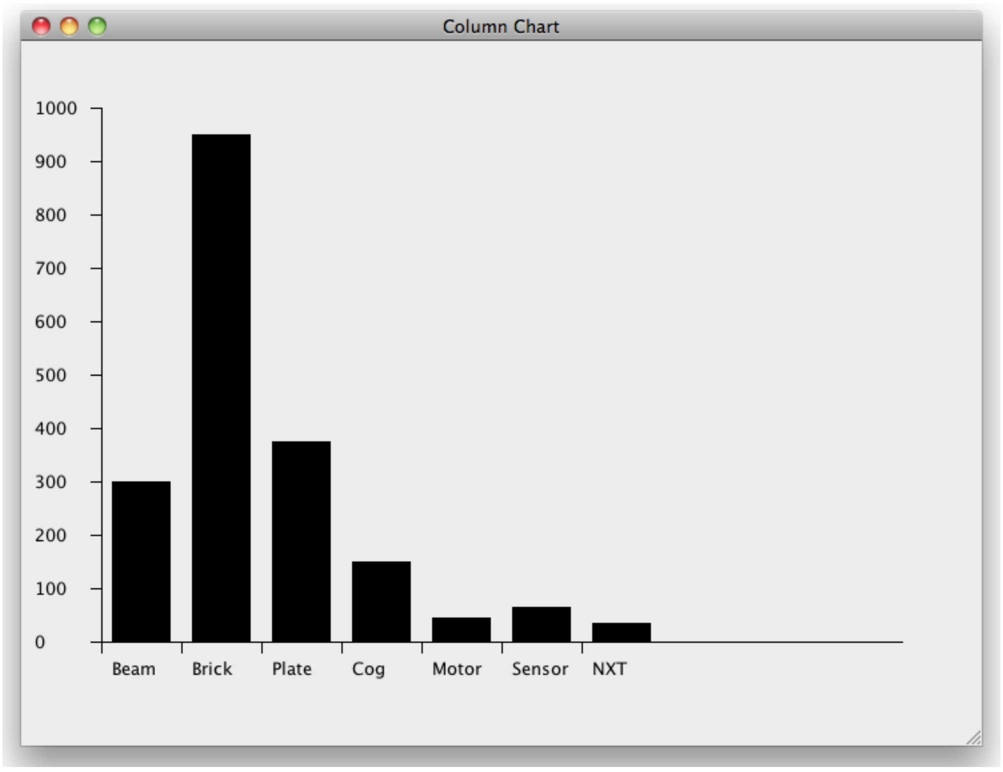

Figure 2: Total parts for all products.

3 Submission instructions

Use the links on the project page of the course website to submit your solutions. You only need to supply your MaSH program code. We will supply all the data files for testing.

4 Marking criteria

General advice as to how the programs will be assessed:

- Programs that are complete and function according to the requirements will get more marks than programs that don't.
- Programs that don't work and/or are incomplete will still attract part marks. Make sure you submit anything you have done.
- Commenting is an important part of quality programming. In particular, we will look for these kinds of comments:
  - a header comment that identifies the programmer, the program and its purpose;
  - comments that describe the purpose of any global variables and constants; and
  - comments that describe the purpose of any methods (other than the standard ones like `main` or `paintWindow`).

  On the other hand it is possible to write too many comments. It is usually not necessary to write comments that just describe what individual statements do. Always assume that the reader of your program knows the programming language at least as well as you do.
- Programs that make good use of methods to organize and simplify programs will be rewarded. As will programs that have good choices of variable names.
- Points will be awarded for appropriate use of constants.