

**DUE: March 27th, 2006**

**Part 1 :**

Submit: Printout for sourceCode and output, Bring Copy of the program.

**Goal:**

The goals of this assignment are to:

1. write your first Java application
2. practice using comments in your code
3. learn how to use command-line input
4. learn how to convert values from String to int
5. practice using mathematical operators
6. learn how to use if-else statements

**Program:**

A robot and a treasure are located somewhere in a room (which is empty except for the robot and the treasure). The floor of the room is covered with square tiles, so you can think of the room as having a giant piece of graph paper covering its floor. The robot and the treasure each occupy a different tile in this grid, and you know the location of each one. The locations are specified using (x,y) coordinates.

Your job is to write a program that will take the coordinates of the robot and the treasure and then determine and provide directions to the robot for how to travel to the treasure. You can assume that the robot begins by facing in the positive Y direction. Your output will say something like "go forward 3 tiles, turn left, go forward 4 tiles".

You enter the coordinates using command-line arguments, as we did in class.

So, your program has to do the following:

a.read 4 parameters from the command line, in the following order:

1. x location of the robot
2. y location of the robot
3. x location of the treasure
4. y location of the treasure

b.echo the input (i.e., output it on the screen);

c.determine and output directions for the robot to travel to the coin.

You can assume that the input values are all whole numbers. Your calculated outputs should be whole numbers in terms of the number of tiles to travel in each direction.

**Sample run.**

Below is a sample run for two cases. The unix command line is highlighted in bold font.

```
unix$ java hw3a 0 0 3 3
```

```
robot is at (0,0)
```

```
treasure is at (3,3)
```

```
go forward 3 tiles, turn right, go forward 3 tiles, stop.
```

```
unix$ java hw3a 4 3 2 1
```

```
robot is at (4,3)
```

```
treasure is at (2,1)
```

```
turn around, go forward 2 tiles, turn right, go forward 2 tiles, stop.
```

**Source code.**

Your source code (i.e., your .java file) must be neat and clearly commented. You must have a header comment and you should comment the end of each block (i.e., each }).

**Part 2 :**

**Submit: Printout sourceCode and output, Bring sourceCode of the program.**

**Goal:**

The goals of this assignment are to:

- ❑ learn how to use Java native classes
- ❑ practice using loops and switch statements

**Program:**

As with the part 1 of assignment...

A robot and a treasure are located somewhere in a room (which is empty except for the robot and the treasure). The floor of the room is covered with square tiles, so you can think of the room as having a giant piece of graph paper covering its floor. The robot and the treasure each occupy a different tile in this grid, and you know the location of each one. The locations are specified using (x,y) coordinates.

For this assignment...

Assume that the x- and y-coordinates of the room range from -5 to 5 (it's a small room). Your job is to write a program that will first randomly choose locations for the robot and the treasure in the room. The user may (optionally) enter a seed for the random number generator on the command line.

Your program must then determine and provide directions to the robot for how to travel to the treasure.

You can assume that the robot begins by facing in the positive Y direction.

Your robot can only move ONE tile at a time, so you need to loop, continually updating the robot's position and giving it new instructions, until it reaches the treasure.

So, your program has to do the following:

Initialize the random number generator. First check if the user entered a random number seed and if so, use that seed; otherwise use the current time to seed the random number generator.

Echo the random seed.

Pick random locations for the robot and the treasure within the room.

Echo the locations of the robot and the treasure.

Loop, incrementally determining and outputting directions for the robot to travel to the treasure, until the robot has found the treasure.

You can assume that the locations are whole numbers. Your calculated outputs should be whole numbers in terms of the number of tiles to travel in each direction.

Hint: keep track of which direction the robot is facing, so that you don't turn too many (or not enough) times! (The robot can only face North, South, East or West; it starts by facing North.)

**NOTE.**

In order to do this assignment, you will need to use the `java.util.Random` and `java.util.Date` classes. See Java API to make the correct use.

**Sample run.**

Below is a sample run for two cases. The unix command line is highlighted in bold font.

Below is a sample run for two cases. The unix command line is highlighted in bold font.

```
unix$ java hw3b
random number seed = 1044058435726
robot is at (-5,5)
treasure is at (-3,0)
turn around, go forward 1 tile
go forward 1 tile
go forward 1 tile
go forward 1 tile
go forward 1 tile
turn left, go forward 1 tile
go forward 1 tile
gotcha!!
unix$ java hw3b 3
random number seed = 3
robot is at (0,0)
```

```
treasure is at (1,-2)
turn around, go forward 1 tile
go forward 1 tile
turn left, go forward 1 tile
gotcha!!
unix$
```

**Source code.**

Your source code (i.e., your `.java` file) must be neat and clearly commented. You must have a header comment and you should comment the end of each block (i.e., each `}`).

**Part 3 :****goal.**

The goals of this assignment are to:

1. learn how to create your own Java classes
2. practice designing and writing object-oriented code

**program.**

This assignment is to re-write the previous assignment using an object-oriented methodology. Re-read part2 so you remember what the end result of the assignment is. Then follow the instructions below to re-write the assignment as an object-oriented program.

1. Create your own public class called **Thing**. Inside the **Thing** class, define the following:
  - four constants representing the directions NORTH, SOUTH, EAST and WEST
  - a private variable that contains a random number generator (i.e., a **Random** object)
  - a private variable that contains the maximum size of a grid coordinate (e.g., 5)
  - two private variables containing the location of an object (i.e., a robot or a treasure) in the grid (i.e., **x** and **y**)
  - a constructor that takes the random number generator object and the maximum grid coordinate as arguments, uses these arguments to initialize instance variables within the new **Thing** object being constructed and initializes the **x** and **y** instance variables to a random location in the grid
  - a method called **pickCoordinate()** which chooses a new random location in the grid (i.e., sets **x** and **y** to random grid coordinates)
  - a method called **toString()** that returns a **String** containing the current values of the **x** and **y** coordinates (e.g., in the form "(x,y)")
  - a method called **getX()** that returns the current value of the **x** coordinate
  - a method called **getY()** that returns the current value of the **y** coordinate
  - a method called **setX()** that takes an integer argument and sets the value of the **x** coordinate to the argument value
  - a method called **setY()** that takes an integer argument and sets the value of the **y** coordinate to the argument value
2. Create your own public class called **Robot** that extends the **Thing** class. Inside the **Robot** class, define the following:
  - four constants representing the types of turns the robot might make: NO\_TURN, LEFT, RIGHT, AROUND
  - a private variable that contains the current direction that the robot is facing (this will take on one of the **Thing** constant values NORTH, SOUTH, EAST or WEST)
  - a constructor that invokes its parent's constructor and also initializes the value of the facing variable to NORTH

- a method called **getFacing()** that returns the value of the facing variable
  - a method called **move()** that takes one argument, the new direction in which the robot should move (i.e., NORTH, SOUTH, EAST or WEST) and returns the type of turn (i.e., NO\_TURN, LEFT, RIGHT, AROUND) that the robot needs to make in order to achieve the new facing; this method should execute the move by setting the value of the facing variable and also adjusting the value of the x or y coordinate (which are defined in the parent class)
  - a method called **printMove()** that takes one argument, the type of turn (i.e., NO\_TURN, LEFT, RIGHT, AROUND) and prints to the screen (using **System.out.println**) the move that the robot is making (e.g., "turn left, go forward 1 tile").
3. A `hw3c<yourname>` class with a **main()** method that initializes and invokes everything else. This method controls the program to work in the same way that `hw2` worked. Here you will declare and instantiate the **Random** number generator object. You'll declare and instantiate a **Robot** object, passing the random object as a parameter. You'll also declare and instantiate a **Thing** object that will act as the treasure. Instantiating a robot and a treasure object will place each in a random location in the grid, by setting the x and y coordinates in the respective constructors. As with `hw2`, calculate the directions for the robot to move to the treasure, one tile at a time. Print the directions, one step at a time, looping until the robot has reached the treasure.

**sample run.**

Below is a sample run for two cases. The unix command line is highlighted in bold font.

```
unix$ java hw3c
random number seed = 1044058435726
robot is at (-5,5)
treasure is at (-3,0)
turn around, go forward 1 tile
go forward 1 tile
go forward 1 tile
go forward 1 tile
go forward 1 tile
go forward 1 tile
turn left, go forward 1 tile
go forward 1 tile
gotcha!!
unix$ java hw3c
random number seed = 3
robot is at (0,0)
treasure is at (1,-2)
turn around, go forward 1 tile
go forward 1 tile
turn left, go forward 1 tile
gotcha!!
unix$
source code.
```

Your source code (i.e., your `.java` file) must be neat and clearly commented. You must have a header comment and you should comment the end of each block (i.e., each `}`).