**CSC262** Programming in JAVA - **Project**

|  |  |
| --- | --- |
| **First Name** | Charles |
| **Last Name** | Kerr |
| **Student ID#** | 041237771 |
| **Email Address** | cwkerr1989@gmail.com |

**Robot with Graphical User Interface**

Table of Contents

[How to submit your work 1](#_Toc16700189)

[Part One – Robot Class – 35 Points 2](#_Toc16700190)

[Part Two - Graphical User Interface – 65 Points 3](#_Toc16700191)

# **How to submit your work**

After filling all the parts in this file, please follow the following steps.

1. Add your name and Student ID# to the first page.
2. Save the file in the original format (Docx or Doc)

(please do not convert to other file formats e.g. PDF, ZIP, RAR, …)

1. Rename the file as

*YOUR First Name YOUR Last Name -* ***Project****.docx*

**Example:**

John Smith - Project.docx

**Please do not seek any assistance when programing this project. You might be asked to explain your code in details. Contact your instructor for any clarifications.**

# Part One – Robot Class – 35 Points

In this project you will create a robot class. The robot objects move within a grid similar to the 3x3 grid shown below to pick up and carry the letters on the grid to different locations.

|  |  |  |
| --- | --- | --- |
| B |  |  |
|  |  | D |
|  | C |  |

The grid size should be declared as constants, use a 25X25 grid for our program.

The Robot class must have the following data members and methods:

1. The **Robot** class has three private data members
   1. An integer type variable for the **x** component of the location of the robot on the grid
   2. An integer type variable for the **y** component of the location of the robot on the grid
   3. A char type variable “**payload**” to hold letters as load
2. The Robot class must have the following public member functions
   1. A **default constructor** places the robot at location (0,0) and payload to character space: ‘ ‘ ( empty)
   2. A **constructor** that receives three parameters to initialize the private data members.
   3. Include a **set** and **get** method for each of the private data members.
   4. Include a member function **print()** that prints the location of the robot on the grid as well as its load.
   5. Method **pickup(int lx, int ly)** with boolean return type. This method examines the current location of the robot and if not at location (lx, ly), it should display a message “not at (lx,ly)” and return false. If the is no load (letters) at the location print “No load at this location” and return false. Otherwise, the method should examine the payload of the robot and if empty it will pick up the load at location (lx,ly) and place it in the payload and returns true. Note, the character at location (lx,ly) on the grid should be set to blank ( space)
   6. Method **dropOff(int lx, int ly)** with Boolean return type. This method examines the current location of the robot and if not at location (lx, ly), it should display a message “not at (lx,ly)” and return false. Otherwise, the method examines the payload of the robot and if full, it will drop its load at location (lx, ly) on the grid and and returns true. Note, the grid element grid[lx][ly] should be set to the character that was placed there.
   7. Method **moveRight()** with a void return type. The method moves robot one unit to the right on the grid. If the robot hits the right boundary of the grid, it should print “Right boundary reached”.
   8. Method **moveLeft()** with a void return type. The method moves robot one unit to the left on the grid. If the robot hits the left boundary of the grid, it should print “Left boundary reached”.
   9. Method **moveUp()** with a void return type. The method moves robot one unit up on the grid. If the robot hits the top boundary of the grid, it should print “Top boundary reached”.
   10. Method **moveDown()** with a void return type. The method moves robot one unit down on the grid. If the robot hits the bottom boundary of the grid, it should print “Bottom boundary reached”.
   11. Method **MoveTo(int lx, int ly)** with boolean return type. This method checks to see whether the location (lx, ly) is within the boundary of the grid, if not returns false. Otherwise, it uses the combination of moveUp(), moveDown(), moveRight(), moveLeft() methods to move the robot to this new location on the grid. Note: if the robot has to move, say 5, units to the right you must write a loop and call moveRight() five times rather that literally writing moveRight() five times.

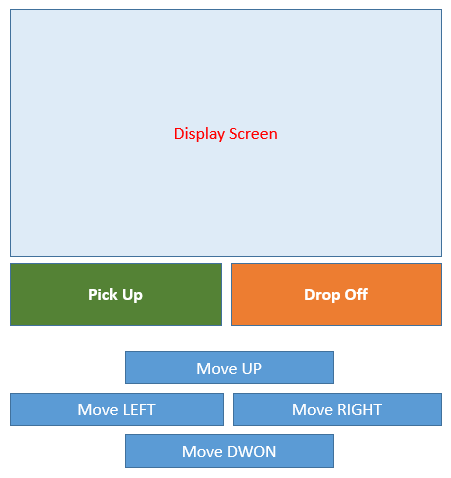
Write a nonmember function **print2D()** that reveries a 2D array of characters and prints it ( in our case 25 elements per row).

Write a main function to test the operation of the robot.

1. You program should define a 2D array representing the grid **char grid [26][26]** **.** Initialize all the elements of the 2D array to blank. Then, place character ‘B’ at location (10, 8) and ‘C’ at (22, 4). Use **print2D()** to print the grid
2. Create two robots R1 and R2 using the two constructors. Print the location of the robots.
3. Use “moveTo()” method to place R1 at location (23, 24) and R2 at location (15, 3). Print the location of the robots
4. Direct R1 to pick up ‘B’ at location (10, 8) and place it at location (20,20). Print the location of the robot. Print the grid.
5. Direct R2 to pick up ‘C’ at location (22, 4) and place it at location (0, 0). Print the location of the robot. Print the grid.

# Part Two - Graphical User Interface – 65 Points

Create a Graphical User Interface for simulating the robot. A sample GUI for this project may be as following.



|  |
| --- |
| **Your C++ code for this HW** |
| //Robot.java (Part 1)  package com.csc262.robotp1;  public class Robot {  public static char grid[][]= new char[25][25];  private int xLocation;  private int yLocation;  private char payload;  Robot() {  xLocation = 0;  yLocation = 0;  payload = ' ';  }  Robot(int x, int y, char letter) {  xLocation = x;  yLocation = y;  payload = letter;  }  int getx() { return xLocation; }  void setx(int newx) { xLocation = newx; }  int gety() { return yLocation; }  void sety(int newy) { yLocation = newy; }  char getPayload() { return payload; }  void setPayload(char newload) { payload = newload;}    void moveLeft(){  xLocation--;  if(xLocation==0)  System.out.println("Left boundary reached.");  }  void moveRight(){  xLocation++;  if(xLocation==24)  System.out.println("Right boundary reached.");  }  void moveUp(){  yLocation--;  if(yLocation==0)  System.out.println("Top boundary reached.");  }  void moveDown(){  yLocation++;  if(yLocation==24)  System.out.println("Bottom boundary reached.");  }    boolean moveTo(int lx, int ly)  {  if (lx < 0 || lx > 24 || ly < 0 || ly > 24) {  System.out.println("Destination is out of bounds!");  return false;  }  else  System.out.printf("Moving to %d,%d... ", lx,ly);    while (xLocation < lx)  {moveRight();}  while (xLocation > lx)  {moveLeft();}  while (yLocation < ly)  {moveDown();}  while (yLocation > ly)  {moveUp();}  System.out.println("Destination reached.");  return true;  }  boolean pickUp(int lx, int ly) {  if (xLocation != lx || yLocation != ly){  System.out.printf("The robot is not at location %f,%f!\n", lx,ly);  return false;  }  if(grid[lx][ly]=='.'){  System.out.printf("No load at location %f,%f to pick up!", lx,ly);  return false;  }  if(payload!=' '){  System.out.println("This robot is already carrying a payload! Drop something off first to make room.");  return false;  }    if (grid[lx][ly] != '.' && payload==' ' && xLocation==lx && yLocation==ly) {  payload = grid[lx][ly];  grid[lx][ly] = '.';  System.out.printf("Payload \"%c\" picked up.\n", payload);  }  return true;  }  boolean dropOff(int lx, int ly) {  if (xLocation != lx || yLocation != ly){  System.out.printf("The robot is not at location %f,%f!\n", lx,ly);  return false;  }  else  if (payload != ' ' && grid[lx][ly] == '.') {  grid[lx][ly] = payload;  System.out.printf("Dropped off load %c at location %d, %d.\n", payload, xLocation, yLocation);  payload = ' ';  }  return true;  }    static void print2D(){  for(int i=0;i<25;i++){  for(int j=0;j<25;j++){  System.out.print(grid[i][j]+" ");  }  System.out.println();  }  }  public static void main(String[] args) {  for(int i=0;i<grid.length;i++)  for(int j=0;j<grid.length;j++)  grid[i][j]='.';    grid[10][8]='B';  grid[22][4]='C';  print2D();    Robot R1=new Robot();  Robot R2=new Robot(3,4,' ');  System.out.printf("Robot 1 location: %d, %d\n", R1.getx(), R1.gety());  System.out.printf("Robot 2 location: %d, %d\n", R2.getx(), R2.gety());  R1.moveTo(23,24);  R2.moveTo(15,3);  System.out.printf("Robot 1 location: %d, %d\n", R1.getx(), R1.gety());  System.out.printf("Robot 2 location: %d, %d\n", R2.getx(), R2.gety());  R1.moveTo(10,8);  R1.pickUp(10,8);  R1.moveTo(20,20);  R1.dropOff(20,20);  System.out.printf("Robot 1 location: %d, %d. Payload: %c\n", R1.getx(), R1.gety(), R1.getPayload());  R2.moveTo(22,4);  R2.pickUp(22,4);  R2.moveTo(0,0);  R2.dropOff(0,0);  System.out.printf("Robot 2 location: %d, %d. Payload: %c\n", R2.getx(), R2.gety(), R2.getPayload());  print2D();  }  }  App.java (Part 2)  package com.csc262.robot;  import javafx.application.Application;  import javafx.fxml.FXMLLoader;  import javafx.scene.Parent;  import javafx.scene.Scene;  import javafx.stage.Stage;  import java.io.IOException;  public class App extends Application {  private static Scene scene;  @Override  public void start(Stage stage) throws IOException {  scene = new Scene(loadFXML("RoboGUI"), 900, 525);  stage.setScene(scene);  stage.show();  }  static void setRoot(String fxml) throws IOException {  scene.setRoot(loadFXML(fxml));  }  private static Parent loadFXML(String fxml) throws IOException {  FXMLLoader fxmlLoader = new FXMLLoader(App.class.getResource(fxml + ".fxml"));  return fxmlLoader.load();  }  public static void main(String[] args) {  launch();  }  }  //RobotController.java  package com.csc262.robot;  import javafx.event.ActionEvent;  import javafx.fxml.FXML;  import javafx.scene.Node;  import javafx.scene.control.Button;  import javafx.scene.control.Label;  import static javafx.scene.input.KeyCode.DOWN;  import static javafx.scene.input.KeyCode.UP;  import static javafx.scene.input.KeyCode.LEFT;  import static javafx.scene.input.KeyCode.RIGHT;  import javafx.scene.input.KeyEvent;  import javafx.scene.layout.GridPane;  public class RobotController {    @FXML  public GridPane grid=new GridPane();  public Node bot;  public Label status;  public Label position;  public Label q;  public Label s;  public Label v;  public Label g;  public Label z;  public Label load;  public boolean cargo=false;  public Button up;  public Button down;  public Button left;  public Button right;    public int x= 0;  public int y= 0;      @FXML  void moveDown(ActionEvent event) {  if (y<24){  grid.setRowIndex(bot,++y);  bot.toFront();  position.setText("Current location: "+x+", "+y);  }  }  @FXML  void moveLeft(ActionEvent event) {  if (x>0)  grid.setColumnIndex(bot,--x);  bot.toFront();  position.setText("Current location: "+x+", "+y);  }  @FXML  void moveRight(ActionEvent event) {  if (x<24)  grid.setColumnIndex(bot,++x);  bot.toFront();  position.setText("Current location: "+x+", "+y);  }  @FXML  void moveUp(ActionEvent event) {  if (y>0)  grid.setRowIndex(bot,--y);  bot.toFront();  position.setText("Current location: "+x+", "+y);  }        @FXML  void pickUp(ActionEvent event){  if(y==grid.getRowIndex(q)&&x==grid.getColumnIndex(q) && !cargo){  load=q;  cargo=true;  grid.getChildren().remove(q);  status.setText("Carrying payload: "+q.getText().charAt(0));  }  else if (y==grid.getRowIndex(s)&&x==grid.getColumnIndex(s)&& !cargo){  load=s;  cargo=true;  grid.getChildren().remove(s);  status.setText("Carrying payload: "+s.getText().charAt(0));  }  else if (y==grid.getRowIndex(v)&&x==grid.getColumnIndex(v)&& !cargo){  load=v;  cargo=true;  grid.getChildren().remove(v);  status.setText("Carrying payload: "+v.getText().charAt(0));  }  else if (y==grid.getRowIndex(g)&&x==grid.getColumnIndex(g)&& !cargo){  load=g;  cargo=true;  grid.getChildren().remove(g);  status.setText("Carrying payload: "+g.getText().charAt(0));  }  else if (y==grid.getRowIndex(z)&&x==grid.getColumnIndex(z)&& !cargo){  load=z;  cargo=true;  grid.getChildren().remove(z);  status.setText("Carrying payload: "+z.getText().charAt(0));  }  }    @FXML  void dropOff(ActionEvent event){    if (cargo){  grid.add(load, x, y);  cargo=false;  status.setText("Carrying payload: None");  }  }    @FXML  public void arrowKeys(KeyEvent event) {  switch (event.getCode()) {  case LEFT:  moveLeft(null);  left.requestFocus();  break;  case RIGHT:  moveRight(null);  right.requestFocus();  break;  case UP:  moveUp(null);  up.requestFocus();  break;  case DOWN:  moveDown(null);  down.requestFocus();  break;  }  }  }  **//RoboGUI.fxml**  <?xml version="1.0" encoding="UTF-8"?>  <?import javafx.scene.control.Button?>  <?import javafx.scene.control.Label?>  <?import javafx.scene.image.Image?>  <?import javafx.scene.image.ImageView?>  <?import javafx.scene.layout.ColumnConstraints?>  <?import javafx.scene.layout.GridPane?>  <?import javafx.scene.layout.Pane?>  <?import javafx.scene.layout.RowConstraints?>  <?import javafx.scene.text.Font?>  <Pane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" onDragDetected="#arrowKeys" onKeyPressed="#arrowKeys" prefHeight="550.0" prefWidth="900.0" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.csc262.robot.RobotController">  <children>  <GridPane fx:id="grid" gridLinesVisible="true" layoutX="30.0" layoutY="35.0" prefHeight="470.0" prefWidth="470.0">  <columnConstraints>  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  </columnConstraints>  <rowConstraints>  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  </rowConstraints>  <children>  <ImageView fx:id="bot" fitHeight="19.0" fitWidth="51.0" pickOnBounds="true" preserveRatio="true">  <image>  <Image url="file:/C:/Users/warren/Documents/NetBeansProjects/Robot/src/botpic.jpg" />  </image>  </ImageView>  <Label fx:id="q" alignment="CENTER" contentDisplay="CENTER" prefHeight="22.0" prefWidth="34.0" text="Q" textAlignment="RIGHT" GridPane.columnIndex="10" GridPane.rowIndex="3">  <font>  <Font size="18.0" />  </font>  </Label>  <Label fx:id="s" alignment="CENTER" contentDisplay="CENTER" prefHeight="22.0" prefWidth="34.0" text="S" textAlignment="RIGHT" GridPane.columnIndex="6" GridPane.rowIndex="9">  <font>  <Font size="18.0" />  </font>  </Label>  <Label fx:id="z" alignment="CENTER" contentDisplay="CENTER" prefHeight="22.0" prefWidth="34.0" text="Z" textAlignment="RIGHT" GridPane.columnIndex="9" GridPane.rowIndex="18">  <font>  <Font size="18.0" />  </font>  </Label>  <Label fx:id="v" alignment="CENTER" contentDisplay="CENTER" prefHeight="22.0" prefWidth="34.0" text="V" textAlignment="RIGHT" GridPane.columnIndex="19" GridPane.rowIndex="8">  <font>  <Font size="18.0" />  </font>  </Label>  <Label fx:id="g" alignment="CENTER" contentDisplay="CENTER" prefHeight="22.0" prefWidth="34.0" text="G" textAlignment="RIGHT" GridPane.columnIndex="18" GridPane.rowIndex="16">  <font>  <Font size="18.0" />  </font>  </Label>  </children>  </GridPane>  <Button fx:id="up" layoutX="604.0" layoutY="73.0" mnemonicParsing="false" onAction="#moveUp" onKeyPressed="#arrowKeys" prefHeight="67.0" prefWidth="169.0" text="Move Up">  <font>  <Font size="18.0" />  </font></Button>  <Button fx:id="down" layoutX="604.0" layoutY="207.0" mnemonicParsing="false" onAction="#moveDown" onKeyPressed="#arrowKeys" prefHeight="67.0" prefWidth="169.0" text="Move Down">  <font>  <Font size="18.0" />  </font></Button>  <Button fx:id="right" layoutX="688.0" layoutY="140.0" mnemonicParsing="false" onAction="#moveRight" onKeyPressed="#arrowKeys" prefHeight="67.0" prefWidth="169.0" text="Move Right">  <font>  <Font size="18.0" />  </font></Button>  <Button fx:id="left" layoutX="518.0" layoutY="140.0" mnemonicParsing="false" onAction="#moveLeft" onKeyPressed="#arrowKeys" prefHeight="67.0" prefWidth="169.0" text="Move Left">  <font>  <Font size="18.0" />  </font></Button>  <Button focusTraversable="false" layoutX="518.0" layoutY="402.0" mnemonicParsing="false" onAction="#pickUp" prefHeight="67.0" prefWidth="169.0" text="Pick Up">  <font>  <Font size="18.0" />  </font>  </Button>  <Button focusTraversable="false" layoutX="687.0" layoutY="402.0" mnemonicParsing="false" onAction="#dropOff" prefHeight="67.0" prefWidth="169.0" text="Drop Off">  <font>  <Font size="18.0" />  </font>  </Button>  <Label fx:id="position" alignment="TOP\_LEFT" focusTraversable="false" layoutX="551.0" layoutY="304.0" prefHeight="17.0" prefWidth="136.0" text="Current Location: 0,0" />  <Label fx:id="status" focusTraversable="false" layoutX="551.0" layoutY="330.0" prefHeight="17.0" prefWidth="136.0" text="Carrying payload: None" />  </children>  </Pane>  /\*  FOR YOUR CONSIDERATION:  A youtube video demonstrating the performance of our robot:  https://www.youtube.com/watch?v=hPCYNJ6Y2Tw  A github repo with project files and resources:  https://github.com/cwkerr1989/CSC262/  The darling robot emoji portraying our fearless protagonist, such that you may place the image in your local directory to recreate our results:  file:/C:/Users/[yourUserName]/Documents/NetBeansProjects/Robot/src/botpic.jpg    \*/ |

Run the code and insert the result in the following box.

|  |
| --- |
| **The result of the query** |
| **Copy and paste the result here (e.g. the screen shot of the result you get by running the code).** |