Appendix: Shortened API documentation for exam of Application Development.

This documentation is based on docs.oracle.com/javase/8/docs/api

java.lang

Class Math

Field Summary

`static double`	**`E`** The `double` value that is closer than any other to *e*, the base of the natural logarithms.
`static double`	**`PI`** The `double` value that is closer than any other to *pi*, the ratio of the circumference of a circle to its diameter.

Method Summary

`static double`	**`abs`**`(double a)` Returns the absolute value of a `double` value.
`static double`	**`cos`**`(double a)` Returns the trigonometric cosine of an angle.
`static double`	**`exp`**`(double a)` Returns Euler's number *e* raised to the power of a `double` value.
`static double`	**`hypot`**`(double x, double y)` Returns sqrt(x^2 +y^2) without intermediate overflow or underflow.
`static double`	**`log`**`(double a)` Returns the natural logarithm (base *e*) of a `double` value.
`static double`	**`pow`**`(double a, double b)` Returns the value of the first argument raised to the power of the second argument.
`static double`	**`random`**`()` Returns a `double` value with a positive sign, greater than or equal to `0.0` and less than `1.0`.
`static int`	**`round`**`(float a)` Returns the closest `int` to the argument.
`static double`	**`sin`**`(double a)` Returns the trigonometric sine of an angle.
`static double`	**`sqrt`**`(double a)` Returns the correctly rounded positive square root of a `double` value.
`static double`	**`tan`**`(double a)` Returns the trigonometric tangent of an angle.

java.awt

Class Graphics

Constructor Summary

protected	**Graphics**() Constructs a new Graphics object.

Method Summary

abstract void	**drawArc**(int x, int y, int width, int height, int startAngle, int arcAngle) Draws the outline of a circular or elliptical arc covering the specified rectangle.
abstract void	**drawLine**(int x1, int y1, int x2, int y2) Draws a line, using the current color, between the points (x1, y1) and (x2, y2) in this graphics context's coordinate system.
abstract void	**drawOval**(int x, int y, int width, int height) Draws the outline of an oval.
void	**drawRect**(int x, int y, int width, int height) Draws the outline of the specified rectangle.
abstract void	**drawString**(AttributedCharacterIterator iterator, int x, int y) Draws the text given by the specified iterator, using this graphics context's current color.
abstract void	**drawString**(String str, int x, int y) Draws the text given by the specified string, using this graphics context's current font and color.
abstract void	**fillArc**(int x, int y, int width, int height, int startAngle, int arcAngle) Fills a circular or elliptical arc covering the specified rectangle.
abstract void	**fillOval**(int x, int y, int width, int height) Fills an oval bounded by the specified rectangle with the current color.
abstract void	**fillRect**(int x, int y, int width, int height) Fills the specified rectangle.
abstract Color	**getColor**() Gets this graphics context's current color.
abstract void	**setColor**(Color c) Sets this graphics context's current color to the specified color.

java.util

Class ArrayList

Constructor Summary

ArrayList()	Constructs an empty list with an initial capacity of ten.
ArrayList(int initialCapacity)	Constructs an empty list with the specified initial capacity.

Method Summary

boolean	**add**(E e) Appends the specified element to the end of this list.
void	**add**(int index, E element) Inserts the specified element at the specified position in this list.
void	**clear**() Removes all of the elements from this list.
boolean	**contains**(Object o) Returns `true` if this list contains the specified element.
E	**get**(int index) Returns the element at the specified position in this list.
int	**indexOf**(Object o) Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
boolean	**isEmpty**() Returns `true` if this list contains no elements.
E	**remove**(int index) Removes the element at the specified position in this list.
boolean	**remove**(Object o) Removes the first occurrence of the specified element from this list, if it is present.
E	**set**(int index, E element) Replaces the element at the specified position in this list with the specified element.
int	**size**() Returns the number of elements in this list.

java.lang

Class Random

Constructor Summary

Random()	Creates a new random number generator.
Random(long seed)	Creates a new random number generator using a single long seed.

Method Summary

int	**nextInt()** Returns next pseudorandom, uniformly distributed int value from this random number generator's sequence.
int	**nextInt(int n)** Returns a pseudorandom, uniformly distributed int value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.
double	**nextDouble()** Returns next pseudorandom, uniformly distributed double value between 0.0 and 1.0 from this random number generator's sequence.

java.lang

Class String

Method Summary

char	**charAt**(int index) Returns the char value at the specified index.
int	**compareTo**(String anotherString) Compares two strings lexicographically.
int	**compareToIgnoreCase**(String str) Compares two strings lexicographically, ignoring case differences.
String	**concat**(String str) Concatenates the specified string to the end of this string.
boolean	**contains**(CharSequence s) Returns true if and only if this string contains the specified sequence of char values.
boolean	**endsWith**(String suffix) Tests if this string ends with the specified suffix.
boolean	**equals**(Object anObject) Compares this string to the specified object.
boolean	**equalsIgnoreCase**(String anotherString) Compares this String to another String, ignoring case considerations.
int	**indexOf**(int ch) Returns the index within this string of the first occurrence of the specified character.
int	**indexOf**(String str) Returns the index within this string of the first occurrence of the specified substring.
boolean	**isEmpty**() Returns true if, and only if, length() is 0.
int	**lastIndexOf**(int ch) Returns the index within this string of the last occurrence of the specified character.
int	**lastIndexOf**(String str) Returns the index within this string of the rightmost occurrence of the specified substring.
int	**length**() Returns the length of this string.
boolean	**startsWith**(String prefix) Tests if this string starts with the specified prefix.
String	**substring**(int beginIndex) Returns a new string that is a substring of this string.
String	**substring**(int beginIndex, int endIndex) Returns a new string that is a substring of this string.
String	**toUppercase**() Converts all of the characters in this String to upper case using the rules of the default locale.

javax.swing

Class JTextField

```
public class JTextField
extends JTextComponent
```

Method Summary

void	**copy**() Transfers the currently selected range in the associated text model to the system clipboard, leaving the contents in the text model.
void	**cut**() Transfers the currently selected range in the associated text model to the system clipboard, removing the contents from the model.
int	**getSelectionEnd**() Returns the selected text's end position.
int	**getSelectionStart**() Returns the selected text's start position.
String	**getText**() Returns the text contained in this TextComponent.
String	**getText**(int offs, int len) Fetches a portion of the text represented by the component.
boolean	**isEditable**() Returns the boolean indicating whether this TextComponent is editable or not.
void	**setEditable**(boolean b) Sets the specified boolean to indicate whether or not this TextComponent should be editable.
void	**setSelectedTextColor**(Color c) Sets the current color used to render the selected text.
void	**setSelectionColor**(Color c) Sets the current color used to render the selection.
void	**setSelectionEnd**(int selectionEnd) Sets the selection end to the specified position.
void	**setSelectionStart**(int selectionStart) Sets the selection start to the specified position.
void	**setText**(String t) Sets the text of this TextComponent to the specified text.

javax.swing

Class JButton

```
public class JButton
extends AbstractButton
```

Method Summary

String	**getText**() Returns the button's text.
void	**setText**(String t) Sets the button's text.
boolean	**isSelected**() Returns the state of the button. True if the toggle button is selected, false if it's not.
void	**setIcon**(Icon defaultIcon) Sets the button's default icon. This icon is also used as the "pressed" and "disabled" icon if there is no explicitly set pressed icon.

java.lang

Class Integer

Method Summary

int	**compareTo**(Integer anotherInteger) Compares two Integer objects numerically.
double	**doubleValue**() Returns the value of this Integer as a double.
boolean	**equals**(Object obj) Compares this object to the specified object.
int	**intValue**() Returns the value of this Integer as an int.
static int	**parseInt**(String s) Parses the string argument as a signed decimal integer.
static int	**parseInt**(String s, int radix) Parses the string argument as a signed integer in the radix specified by the second argument.
String	**toString**() Returns a String object representing this Integer's value.
static String	**toString**(int i) Returns a String object representing the specified integer.
static String	**toString**(int i, int radix) Returns a string representation of the first argument in the radix specified by the second argument.

java.lang

Class Double

Method Summary

int	**compareTo**(Double anotherDouble) Compares two Double objects numerically.
double	**doubleValue**() Returns the double value of this Double object.
boolean	**equals**(Object obj) Compares this object to the specified object.
int	**intValue**() Returns the value of this Double as an int (by casting to type int).
static double	**parseDouble**(String s) Returns a new double initialized to the value represented by the specified String, as performed by the valueOf method of class Double.
String	**toString**() Returns a string representation of this Double object.
static String	**toString**(double d) Returns a string representation of the double argument.

Explorer.h

```
/*
Declaration of the Explorer class used for the robot.
*/

/* Definition of wheel diameter and track width of the robot in
cm: */
#define WHEEL_DIAM 4.96
#define TRACKWIDTH 14

/* Arduino pins used for the ultrasonic sensor: */
#define TRIGGER_PIN  3
#define ECHO_PIN     5

/* Maximum distance we want to ping for (in centimeters): Maximum
sensor distance is rated at 400-500cm. */
#define MAX_DISTANCE 300

#include "Arduino.h"
#include <Wire.h>
#include <EVShield.h>
#include <EVs_NXTTouch.h>
#include <NewPing.h>

/**
  @brief This class interfaces with EVShield to create an explorer
robot.
  Setup:
    - Two motors to drive, connected to Bank A of the EVShield
    - One motor on which the ultrasonic sensor is mounted: M1 on
Bank B
    - Touch sensor on port BBS2 (Bank B)
*/

class Explorer {
// class variables:
private:
// pointers to objects created in main sketch:
  EVShield * evshield;
  NewPing * sonar;
  EVs_NXTTouch * touch;
  String instructions = "";

public:
  bool isDriving = false; // false=not driving, true=driving
  bool forward = true; // we start driving forward

/* Initial motor direction: (depends on how motors are mounted) */
  SH_Direction motor_direction = SH_Direction_Reverse;
  int speed = SH_Speed_Slow; // SH_Speed_Medium

/* first motor (M1) on Bank B on which the ultrasonic sensor is
mounted: */
  SH_Motor sensorMotor = SH_Motor_1;

// methods:
  void init(EVShield * s, NewPing * p, EVs_NXTTouch * t);

// driving related methods:
  void reverseDirection();
  void drive(int distance = 0);
  void stop();
  void turn(int angle);

// sensor related methods:
  void checkSensors();
  unsigned int readDistance(); // read distance from
                               // ultrasonic sensor

// extra methods:
  void reverseTurn();
  void find_a_way_out();

// methods to process instructions:
  void readCommand(String input);
  void doCommand(unsigned int count, String number, char command);
  void setInstructions(String s);
};
```