Test 6

All answers should be written on these first two pages.

This test is working Java and Android code. Feel free to keep pages 3 - 12 and run the code to verify your answers.

**Problem 1:**

**Problem 2:**

**Problem 3:**

**Problem 4:**

**Problem 5:**

**Problem 6:**

**Problem 7:**

**Problem 8:**

**Problem 9:**

**Problem 10:**

**Problem 11:**

**Problem 12:**

For problems 13 – 16, you must document the transitions through the state machine

**Problem 13:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Current State** | **Input** | **Next State** | **Output** |
| **Off** |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Problem 14:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Current State** | **Input** | **Next State** | **Output** |
| **Off** |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Problem 15:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Current State** | **Input** | **Next State** | **Output** |
| **Off** |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Problem 16:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Current State** | **Input** | **Next State** | **Output** |
| **Off** |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Problem 17:**

**Problem 18:**

**Problem 19:**

**Problem 20:**

package nyc.c4q;

public class FunctionsEatingFunctions {

public static class StringBuilder{

private String value;

public StringBuilder(String start){

value = start;

}

public StringBuilder append(String other){

return new StringBuilder(value + other);

}

public String get(){

return value;

}

}

**public void problem1() {**

StringBuilder builder = new StringBuilder("start");

builder.append("1");

builder.append("2");

builder.append("3");

System.out.println(builder.get());

**}**

**public void problem2() {**

StringBuilder builder = new StringBuilder("start").append("1");

StringBuilder builder2 = new StringBuilder("end");

builder2.append("2");

System.out.println(builder.get());

System.out.println(builder2.get());

**}**

**public void problem3() {**

StringBuilder builder = new StringBuilder("start").append("1");

builder = builder.append("2");

System.out.println(builder.get());

**}**

**public void problem4() {**

System.out.println(

new StringBuilder("end")

.append("2")

.get()

);

**}**

}

package nyc.c4q;

public class ListManipulations {

private int[] mColors = { 5, 10, 15, 20 };

private int[] mStates = { 40, 50, 60, 70 };

private int[] mTables = { 1, 2, 3, 4 };

**public void problem5() {**

for ( int index1 = 0; index1 < mTables.length; index1++) {

for ( int index2 = 0; index2 < mColors.length; index2++) {

String out = mColors[index1] + ":" + mTables[index2];

System.out.println(out);

}

}

**}**

**public void problem6() {**

int index1 = 0;

int index2 = 1;

while (mStates.length > index1) {

System.out.println(mStates[index1]);

index1 += index2;

index2 += 1;

}

**}**

**public void problem7() {**

for (int index = 0; index < 3; index+=2) {

System.out.println(mColors[index]);

}

**}**

**public void problem8() {**

int[] d1 = new int[3];

for (int index = 0; index < d1.length; index++) {

d1[index] = mStates[index];

}

int[] d2 = new int[4];

for (int index = 0; index < d2.length; index++) {

d2[index] = mColors[index];

}

for (int index = 0; index < d1.length; index++) {

System.out.println(d2[index]);

System.out.println(d1[index]);

}

**}**

}

package nyc.c4q;

public class OptionalOperations {

private static class Optional<T> {

T value;

public Optional(T value){

this.value = value;

}

public T orElse(T other){

if (value != null) {

return value;

}

return other;

}

public Optional<T> transform(boolean toTransform, T value){

if (toTransform) {

this.value = value;

}

return this;

}

public Optional<T> filter(T other){

if (value != null){

return this;

}

return new Optional<T>(other);

}

public T get(){

if ( value == null) {

throw new IllegalArgumentException("null value");

}

return value;

}

}

**private static void problem9() {**

Optional<String> maybeString = new Optional<String>("hello");

String result = maybeString.transform(false, "hello2")

.transform(true, "hello1")

.filter("notCrazy")

.get();

System.out.println(result);

**}**

**private static void problem10() {**

Optional<String> maybeString = new Optional<String>(null);

String result = maybeString.transform(false, "hello2")

.transform(false, "hello1")

.filter("notCrazy")

.get();

System.out.println(result);

**}**

**private static void problem11() {**

Optional<String> maybeString = new Optional<String>(null);

String result = maybeString.transform(false, "hello2")

.transform(false, "hello1")

.filter(null)

.filter("test")

.get();

System.out.println(result);

**}**

**private static void problem12() {**

Optional<Integer> maybeString = new Optional<Integer>(null);

int result = maybeString.transform(false, new Integer(5))

.transform(false, new Integer(8))

.filter(null)

.orElse(new Integer(10));

System.out.println(result);

**}**

}

package nyc.c4q;

public class StateMachine {

public enum Input {

TURN\_KEY,

SHIFT\_UP,

SHIFT\_DOWN,

NONE

}

public enum State {

OFF,

IGNITION\_TURNED,

ENGINE\_STARTING,

ENGINE\_IDLE,

FIRST\_GEAR,

SECOND\_GEAR,

REVERSE,

ENGINE\_STOPPING

}

public static State getNextState(State currentState) {

switch (currentState) {

case OFF:

break;

case IGNITION\_TURNED:

return State.ENGINE\_STARTING;

case ENGINE\_STARTING:

return State.ENGINE\_IDLE;

case ENGINE\_IDLE:

return State.FIRST\_GEAR;

case FIRST\_GEAR:

return State.FIRST\_GEAR;

case SECOND\_GEAR:

return State.SECOND\_GEAR;

case REVERSE:

return State.REVERSE;

case ENGINE\_STOPPING:

break;

default:

throw new IllegalArgumentException("invalid state ");

}

return State.OFF;

}

public static void handleState(State currentState) {

switch (currentState) {

case OFF:

break;

case IGNITION\_TURNED:

System.out.println("Ignition");

break;

case ENGINE\_STARTING:

System.out.println("Starting");

break;

case ENGINE\_IDLE:

System.out.println("Idle");

break;

case FIRST\_GEAR:

System.out.println("1st");

break;

case SECOND\_GEAR:

System.out.println("2nd");

break;

case REVERSE:

System.out.println("Reversing");

break;

case ENGINE\_STOPPING:

System.out.println("Stopping");

}

}

public void run(State initialState, Input[] inputs){

State currentState = initialState;

for (int index = 0; index < inputs.length; index++) {

State nextState = getNextState(currentState);

nextState = handleInput(currentState,

nextState,

inputs[index]);

handleState(currentState);

currentState = nextState;

}

}

private State handleInput(State currentState,

State nextState,

Input input) {

if (currentState == State.OFF && input == Input.TURN\_KEY){

return State.IGNITION\_TURNED;

} else if (currentState == State.IGNITION\_TURNED &&

input != Input.NONE){

return State.IGNITION\_TURNED;

} else if (currentState == State.FIRST\_GEAR &&

input == Input.TURN\_KEY){

return State.ENGINE\_STOPPING;

} else if (currentState == State.FIRST\_GEAR &&

input == Input.SHIFT\_DOWN){

return State.REVERSE;

} else if (currentState == State.FIRST\_GEAR &&

input == Input.SHIFT\_UP){

return State.SECOND\_GEAR;

} else if (currentState == State.SECOND\_GEAR &&

input == Input.TURN\_KEY){

return State.ENGINE\_STOPPING;

} else if (currentState == State.SECOND\_GEAR &&

input == Input.SHIFT\_DOWN){

return State.FIRST\_GEAR;

} else if (currentState == State.REVERSE &&

input == Input.TURN\_KEY){

return State.ENGINE\_STOPPING;

} else if (currentState == State.REVERSE &&

input == Input.SHIFT\_UP){

return State.FIRST\_GEAR;

}

return nextState;

}

public static void main(String[] args) {

Input[] inputs = **// Problems 13 - 16**

new StateMachine().run(State.OFF, inputs);

}

**private static Input[] problem13() {**

return new Input[]{

Input.NONE,

Input.NONE,

Input.TURN\_KEY,

Input.NONE,

Input.NONE,

Input.NONE,

Input.TURN\_KEY,

Input.NONE,

};

**}**

**private static Input[] problem14() {**

return new Input []{

Input.TURN\_KEY,

Input.SHIFT\_DOWN,

Input.SHIFT\_UP,

Input.TURN\_KEY,

Input.NONE,

};

**}**

**private static Input [] problem15() {**

return new Input []{

Input.TURN\_KEY,

Input.NONE,

Input.SHIFT\_DOWN,

Input.SHIFT\_UP,

Input.TURN\_KEY,

Input.NONE,

};

**}**

**private static Input [] problem16() {**

return new Input []{

Input.TURN\_KEY,

Input.NONE,

Input.NONE,

Input.SHIFT\_DOWN,

Input.NONE,

Input.NONE,

Input.SHIFT\_UP,

Input.TURN\_KEY,

Input.NONE,

};

**}**

}

package nyc.c4q;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

public class MainActivity extends AppCompatActivity {

private static final int DEFAULT\_VIEW = 1;

private static final String VIEW\_EXTRA = "intent.extra.view";

private static final int BLUE = 0xFF0000FF;

private static final int BLACK = 0xFF000000;

private View mRoot;

private View mButton;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

int viewToDisplay = getIntent()

.getIntExtra(VIEW\_EXTRA, DEFAULT\_VIEW);

setupView(viewToDisplay);

mRoot = findViewById(R.id.main\_view);

if (viewToDisplay == DEFAULT\_VIEW){

mButton = findViewById(R.id.button1);

} else {

mButton = findViewById(R.id.button2);

}

mRoot.setOnClickListener(getClickListener());

mButton.setOnClickListener(getClickListener());

}

private View.OnClickListener getClickListener() {

return new View.OnClickListener() {

@Override

public void onClick(View view) {

if ( view == mRoot) {

mRoot.setBackgroundColor(BLUE);

} else {

mRoot.setBackgroundColor(BLACK);

}}};

}

public void setupView(int viewOptions) {

if (viewOptions == DEFAULT\_VIEW){

setContentView(R.layout.activity\_main);

} else {

setContentView(R.layout.activity\_submenu);

}

}

}

**getIntent**

[Intent](https://developer.android.com/reference/android/content/Intent.html) getIntent ()

Return the intent that started this activity.

**getIntExtra**

int getIntExtra ([String](https://developer.android.com/reference/java/lang/String.html) name, int defaultValue)

Returns the value of an item that previously added with putExtra() or the default value if none was found.

Name String: The name of the desired item

defaultValue int: the value to be returned if no value of the desired type is stored with the given name.

**Problem 17:**

If the currently displayed layout is activity\_main, what value was returned by getIntExtra?

**Problem 18:**

If mButton is set to the view with id R.id.button2, what value was returned by getIntExtra?

**Problem 19:**

Which view must be clicked to turn the background Black?

**Problem 20:**

What is the return value from mRoot.getOnClickListener().equals(mButton.getOnClickListener())?