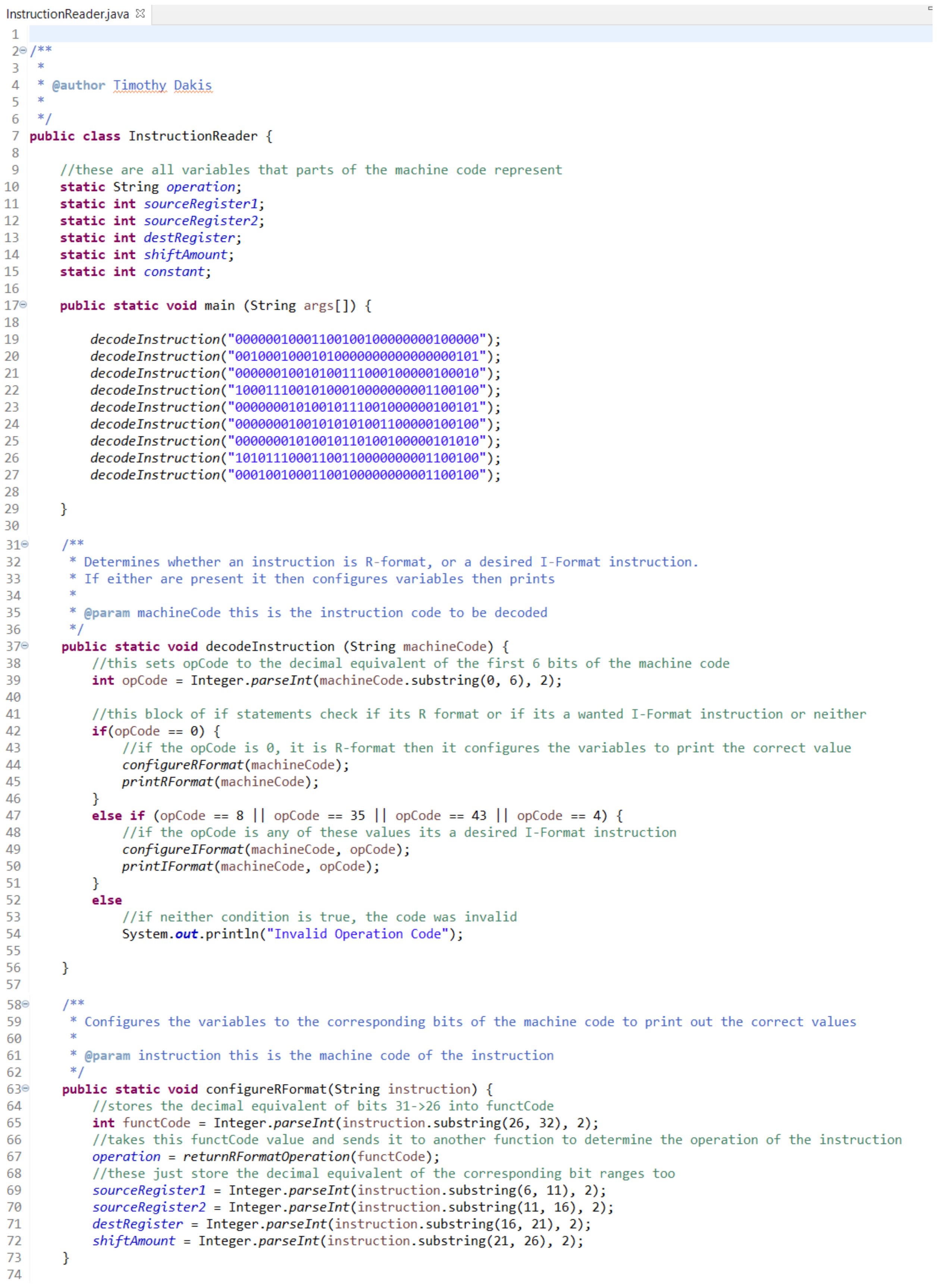
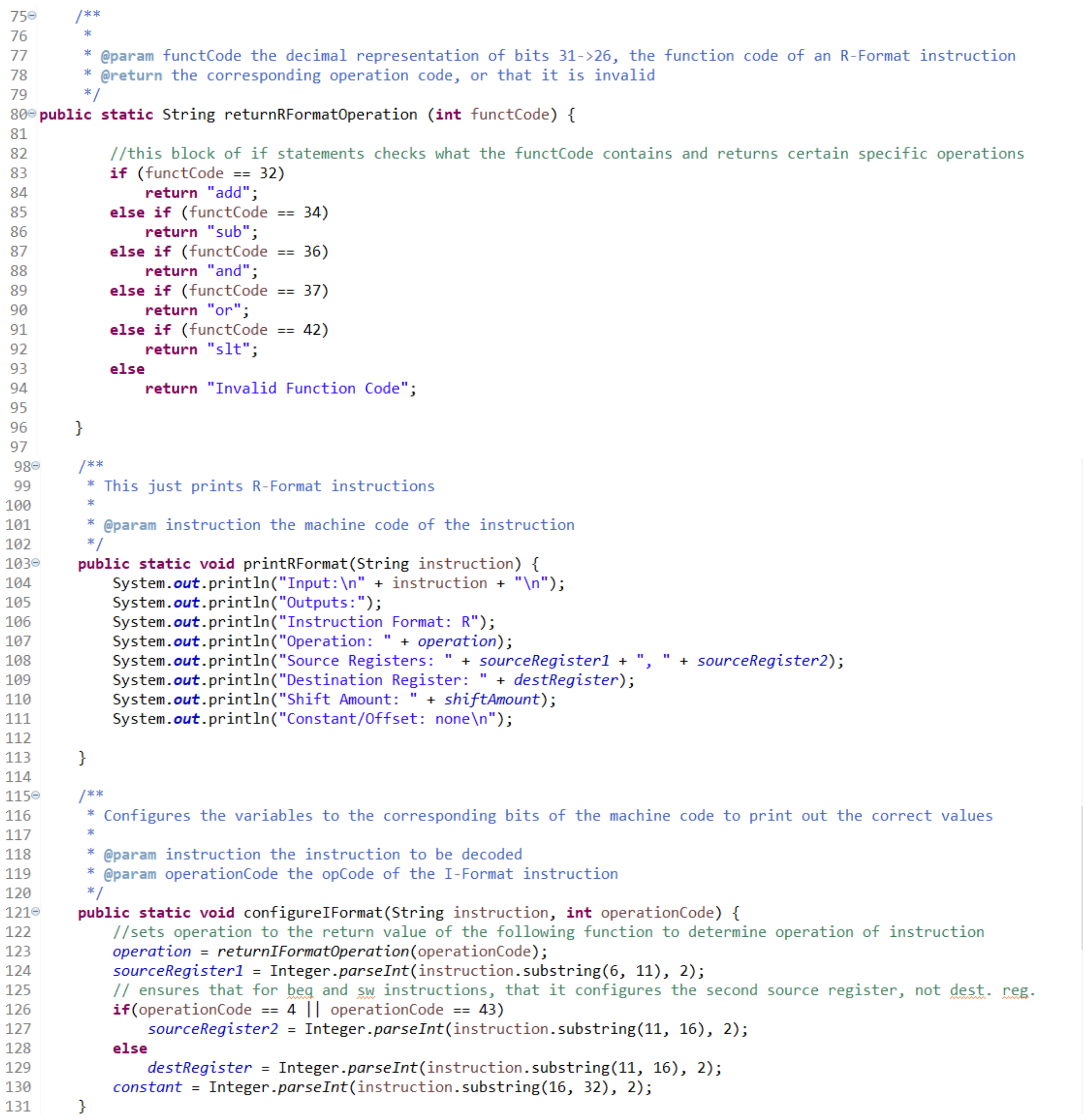


```java

/**
 *
 * @author Timothy Dakis
 *
 */
public class InstructionReader {

    //these are all variables that parts of the machine code represent
    static String operation;
    static int sourceRegister1;
    static int sourceRegister2;
    static int destRegister;
    static int shiftAmount;
    static int constant;

    public static void main (String args[]) {

        decodeInstruction("00000010001100100100000000100000");
        decodeInstruction("00100010001010000000000000000101");
        decodeInstruction("00000010010100111000100000100010");
        decodeInstruction("10001110010100010000000001100100");
        decodeInstruction("00000001010010111001000000100101");
        decodeInstruction("00000001001010101001100000100100");
        decodeInstruction("00000001010010110100100000101010");
        decodeInstruction("10101110001100110000000001100100");
        decodeInstruction("00010010001100100000000001100100");

    }

    /**
     * Determines whether an instruction is R-format, or a desired I-Format instruction.
     * If either are present it then configures variables then prints
     *
     * @param machineCode this is the instruction code to be decoded
     */
    public static void decodeInstruction (String machineCode) {
        //this sets opCode to the decimal equivalent of the first 6 bits of the machine code
        int opCode = Integer.parseInt(machineCode.substring(0, 6), 2);

        //this block of if statements check if its R format or if its a wanted I-Format instruction or neither
        if(opCode == 0) {
            //if the opCode is 0, it is R-format then it configures the variables to print the correct value
            configureRFormat(machineCode);
            printRFormat(machineCode);
        }
        else if (opCode == 8 || opCode == 35 || opCode == 43 || opCode == 4) {
            //if the opCode is any of these values its a desired I-Format instruction
            configureIFormat(machineCode, opCode);
            printIFormat(machineCode, opCode);
        }
        else
            //if neither condition is true, the code was invalid
            System.out.println("Invalid Operation Code");

    }

    /**
     * Configures the variables to the corresponding bits of the machine code to print out the correct values
     *
     * @param instruction this is the machine code of the instruction
     */
    public static void configureRFormat(String instruction) {
        //stores the decimal equivalent of bits 31->26 into functCode
        int functCode = Integer.parseInt(instruction.substring(26, 32), 2);
        //takes this functCode value and sends it to another function to determine the operation of the instruction
        operation = returnRFormatOperation(functCode);
        //these just store the decimal equivalent of the corresponding bit ranges too
        sourceRegister1 = Integer.parseInt(instruction.substring(6, 11), 2);
        sourceRegister2 = Integer.parseInt(instruction.substring(11, 16), 2);
        destRegister = Integer.parseInt(instruction.substring(16, 21), 2);
        shiftAmount = Integer.parseInt(instruction.substring(21, 26), 2);
    }

```



```java
    /**
     *
     * @param functCode the decimal representation of bits 31->26, the function code of an R-Format instruction
     * @return the corresponding operation code, or that it is invalid
     */
public static String returnRFormatOperation (int functCode) {

        //this block of if statements checks what the functCode contains and returns certain specific operations
        if (functCode == 32)
            return "add";
        else if (functCode == 34)
            return "sub";
        else if (functCode == 36)
            return "and";
        else if (functCode == 37)
            return "or";
        else if (functCode == 42)
            return "slt";
        else
            return "Invalid Function Code";

    }

    /**
     * This just prints R-Format instructions
     *
     * @param instruction the machine code of the instruction
     */
    public static void printRFormat(String instruction) {
        System.out.println("Input:\n" + instruction + "\n");
        System.out.println("Outputs:");
        System.out.println("Instruction Format: R");
        System.out.println("Operation: " + operation);
        System.out.println("Source Registers: " + sourceRegister1 + ", " + sourceRegister2);
        System.out.println("Destination Register: " + destRegister);
        System.out.println("Shift Amount: " + shiftAmount);
        System.out.println("Constant/Offset: none\n");

    }

    /**
     * Configures the variables to the corresponding bits of the machine code to print out the correct values
     *
     * @param instruction the instruction to be decoded
     * @param operationCode the opCode of the I-Format instruction
     */
    public static void configureIFormat(String instruction, int operationCode) {
        //sets operation to the return value of the following function to determine operation of instruction
        operation = returnIFormatOperation(operationCode);
        sourceRegister1 = Integer.parseInt(instruction.substring(6, 11), 2);
        // ensures that for beq and sw instructions, that it configures the second source register, not dest. reg.
        if(operationCode == 4 || operationCode == 43)
            sourceRegister2 = Integer.parseInt(instruction.substring(11, 16), 2);
        else
            destRegister = Integer.parseInt(instruction.substring(11, 16), 2);
        constant = Integer.parseInt(instruction.substring(16, 32), 2);
    }

    /**
     * Determines the I-Format operation used
     *
     * @param operationCode opCode of the machine code
     * @return the operation of the instruction, or that the opCode was invalid
     */
    public static String returnIFormatOperation(int operationCode) {
        if(operationCode == 8)
            return "addi";
        else if(operationCode == 35)
            return "lw";
        else if(operationCode == 43)
            return "sw";
        else if(operationCode == 4)
            return "beq";
        else
            return "Invalid I-Format OpCode";
    }

    /**
     * This just prints out I-Format instructions
     *
     * @param instruction the machine code of the instruction
     * @param operationCode the opCode of the instruction
     */
    public static void printIFormat (String instruction, int operationCode) {

        System.out.println("Input:\n" + instruction + "\n");
        System.out.println("Outputs:");
        System.out.println("Instruction Format: I");
        System.out.println("Operation: " + operation);
        //this chain of if statements changes what is printed based on if the instruction is beq or sw, or neither
        if(operationCode == 43 || operationCode == 4) {
            System.out.println("Source Registers: " + sourceRegister1 + ", " + sourceRegister2);
            System.out.println("Destination Register: none");
        }
        else {
            System.out.println("Source Registers: " + sourceRegister1);
            System.out.println("Destination Register: " + destRegister);
        }
        System.out.println("Shift Amount: none");
        System.out.println("Constant/Offset: " + constant +"\n");


    }


}

```

```
Input:
00000010001100100100000000100000

Outputs:
Instruction Format: R
Operation: add
Source Registers: 17, 18
Destination Register: 8
Shift Amount: 0
Constant/Offset: none

Input:
00100010001010000000000000000101

Outputs:
Instruction Format: I
Operation: addi
Source Registers: 17
Destination Register: 8
Shift Amount: none
Constant/Offset: 5

Input:
00000010010100111000100000100010

Outputs:
Instruction Format: R
Operation: sub
Source Registers: 18, 19
Destination Register: 17
Shift Amount: 0
Constant/Offset: none
```

```
Input:
10001110010100010000000001100100

Outputs:
Instruction Format: I
Operation: lw
Source Registers: 18
Destination Register: 17
Shift Amount: none
Constant/Offset: 100

Input:
00000001010010111001000000100101

Outputs:
Instruction Format: R
Operation: or
Source Registers: 10, 11
Destination Register: 18
Shift Amount: 0
Constant/Offset: none

Input:
00000001001010101001100000100100

Outputs:
Instruction Format: R
Operation: and
Source Registers: 9, 10
Destination Register: 19
Shift Amount: 0
Constant/Offset: none
```

```
Input:
00000001010010110100100000101010

Outputs:
Instruction Format: R
Operation: slt
Source Registers: 10, 11
Destination Register: 9
Shift Amount: 0
Constant/Offset: none

Input:
10101110001100110000000001100100

Outputs:
Instruction Format: I
Operation: sw
Source Registers: 17, 19
Destination Register: none
Shift Amount: none
Constant/Offset: 100

Input:
00010010001100100000000001100100

Outputs:
Instruction Format: I
Operation: beq
Source Registers: 17, 18
Destination Register: none
Shift Amount: none
Constant/Offset: 100
```