```java
import java.util.Scanner;

public class TheCache {

	//Properties
	public static final short MAIN_MEMORY = 2048;
	public static final short MAX_BYTE = 0xff;
	public final short SIZE = 16;
	public temp[] node = new temp[SIZE];

	private String input;
	private short slot,tag = -1;
	private boolean valid = false;
	private short[] memory = new short[MAIN_MEMORY];
	private short[] data = new short[SIZE];



	//Constructors
	public TheCache(){

	}


	//Program
	public static void main(String args[]) {

		//get out of static land
		TheCache me = new TheCache();
		me.initializeCache();
		me.doIt();

	}


	//The Program
	public void doIt() {

		//Initialize input, ask for command
		Scanner keyboard = new Scanner(System.in);
		System.out.println("\nWhat would you like to do? \n(R)ead, (W)rite, or (D)isplay Cache");

		input = keyboard.next();

		//Recognize command
		if (input.equalsIgnoreCase("r")) {

			System.out.println("What address would you like to read?");
			short value = (short) keyboard.nextShort(16);
			read(value);
			doIt();

		} else if (input.equalsIgnoreCase("w")) {
			System.out.println("What address would you like to write?");
			short address = keyboard.nextShort(16);
			System.out.println("What data would you like to write at that address?");
			short data = keyboard.nextShort(16);
			write(address, data);
			doIt();

		} else if (input.equalsIgnoreCase("d")) {

			display();
			doIt();

		} else {

			System.out.println("Wrong input..Try again");
```

```
                doIt();

        }

}

//Read value and see if it exists
public boolean read(short address) {
        short tag = (short) ((address & 0xF00) >>> 8);
        short slot = (short) ((address & 0x0F0) >>> 4);
        short offset = (short) (address & 0x00F);

        if (tag == this.node[slot].getTag()) {

                if (this.node[slot].getValid()) {
                        System.out.printf("At that byte there is the value  %x (Cache Hit)\n",
                                        this.node[slot].getData(offset));
                        return true;
                }

                else {

                        this.data(address);
                        System.out.printf("At that byte there is the value %x (Cache Miss)\n",
                                        this.node[slot].getData(offset));
                        return true;
                }
        } else {

                if (!this.node[slot].getValid()) {

                        this.data(address);
                        System.out.printf("At that byte there is the value %x (Cache Miss)\n",
                                        this.node[slot].getData(offset));
                        return true;

                } else {

                        this.data(address);
                        System.out.printf("At that byte there is the value %x  (Cache Miss)\n",
                                        this.node[slot].getData(offset));
                        return true;
                }
        }
}

//Initialize Cache
public void initializeCache() {
        for (short s = 0; s < MAIN_MEMORY; s++) {
                this.memory[s] = (short) (s & 0Xff);
        }

        for (byte b = 0; b < SIZE; b++) {
                this.node[b] = new temp(b);
        }
}

//Setup Data
private void data(int address) {
        short tag = (short) ((address & 0xF00) >>> 8);
        short slot = (short) ((address & 0x0F0) >>> 4);
        short start = (short) (address & 0xFF0);
        short last = (short) (start + SIZE); short value = 0;
        for (short i = start; i < last; i++) {
                this.node[slot].setData(value++, this.memory[i]);
        }
        this.node[slot].setTag(tag);
        this.node[slot].setValid(true);
}
```

```
//Set up the Write
public boolean write( short address, short value ) {
        short tag = (short) ( (address & 0xF00) >>> 8 );
        short slot = (short) ( (address & 0x0F0) >>> 4 );
        short offset = (short) (address & 0x00F);

        if ( tag == this.node[slot].getTag() ) {

                if ( this.node[slot].getValid() ) {

                        this.node[slot].setData(offset, value);
                        System.out.printf( "The value %x has been written to address %x (Cache Hit)\n",
                        this.node[slot].getData(offset), address );
                        return true;

                } else {

                        this.data(address);
                        this.node[slot].setData(offset, value);
                        System.out.printf( "The value %x has been written to address %x (Cache Miss)\n",
                                        this.node[slot].getData(offset), address );
                        return true;

                }
        } else {

                this.data(address);
                this.node[slot].setData(offset, value);
                System.out.printf( "The value %X has been written to address %X (Cache Miss)\n",
                                this.node[slot].getData(offset), address );
                return true;
        }
}

public void display() {
        System.out.println( "Slot Valid Tag       Data" );
        for (byte b = 0; b < SIZE; b++) {
                this.node[b].printCache();

        }
        System.out.println();
}

public short getData(short data) {
        return this.data[data];
}
public void setData(short data, short count) {
        this.data[data] = count;
}
public boolean getValid() {
        return this.valid;

}
public void setValid(boolean valid) {
        this.valid = valid;
}
public void setTag(short tag) {
        this.tag = tag;
}
public short getTag() {
        return this.tag;
}
}
```