CSCI 2302

Objects & Classes Chapter

Creating Object Lab

Using the UML diagram created in the UML Diagram Lab, implement the Java code. After defining the code, implement a “test” program.

Complete the following in the test program:

* instantiate an object
* change one of the states
* print the value of that state
* invoke the additional behavior

|  |
| --- |
| **UML\_DiagramIntoCode** |
| -state1: int  -state2: char  -state3: String  -state4: String = “initial value”  -word: String |
| + UML\_DiagramIntoCode()  + UML\_DiagramIntoCode(state1: int, state2: char, state3: String, state4: String)  +setState1(state1: int): void  +getState1(): int  +setState2(state2: char): void  +getState2(): char  +setState3(state3: String): void  +getState3(): String  +setState4(state4: String): void  +getState4(): String  +method1(word: String): void  +method2(): String  +toString(): String |

public class UML\_DiagramIntoCode {

// list of states - the variables

private int state1;

private char state2;

private String state3;

// if they have an initial state you can assign that here or in the

default constructor

// for example:

private String state4 = "initial value";

private String word;

// default constructor - should be listed first

public UML\_DiagramIntoCode(){

// the default constructor does not take in any variables

// it can assign default - or initial - values here or when the

variable is declared

// for example

this.state1 = 1;

}// end UML\_DiagramIntoCode default constructor

// constructor that accepts all the variables

public UML\_DiagramIntoCode(int state1, char state2, String state3, String state4){

this.state1 = state1;

this.state2 = state2;

this.state3 = state3;

this.state4 = state4;

}// end UML\_DiagramIntoCode constructor that accepts all the variables

// constructor that accepts some variables

// please note that you can have as many constructors as needed –

you see fit -

// having a class that is user friendly is the goal

public UML\_DiagramIntoCode(char state2, String state3){

this.state2 = state2;

this.state3 = state3;

}// end UML\_DiagramIntoCode constructor that accepts some variables

// once you have completed all the constructors, do the setters &

getters

/\* It is your choice if you want to do all the setters then the

getters or do the setters first, then do the getters.

\*/

public void setState1 (int state1){

this.state1 = state1;

}

public int getState1(){

return this.state1;

}

public void setState2(char state2){

this.state2 = state2;

}

public char getState2(){

return this.state2;

}

public void setState3(String state3){

this.state3 = state3;

}

public String getState3(){

return this.state3;

}

public void setState4(String state4){

this.state4 = state4;

}

public String getState4(){

return this.state4;

}

// after the setters and getters, layout the methods for the

object/class

public void method1(String word){

System.out.println("a method with a value given " + word);

}

public String method2(){

return "another method";

}

// finish with the toString method for easy access of the

object/class data

public String toString(){

return "The \_object\_ has " + this.state1 + " and " + this.state2 + " and " + this.state3 + " and " + this.state4;

}

}// end of UML\_DiagramIntoCode

public class TestProgram{

public static void main(String [] args){

UML\_DiagramIntoCode obj = new UML\_DiagramIntoCode();

obj.setState2(‘q’);

System.out.println(obj.getState2());

Obj.method2();

}// end main

}// end TestProgram