• Introduction: Abstract Data Type (ADT)
  • An abstract data type is commonly known as a class of objects
  • An abstract data type in a program is used to represent (the behavior) of some class of object in the real world
  • In this lab, we will write the definition of an abstract data type (called **Roulette** ) that represents Roulette tables in the real world and their behavior.

• Roulette Table
  • Note: A computer program will never be able to represent a real Roulette table. What a computer program can do is represent the functionality of a Roulette table. So we need to know how a Roulette table is used.

• The Roulette Table:
  • Roulette consists of a wheel and a betting area:
  
  ![Roulette Wheel and Table]

• How to play roulette:
  • People place bets in the betting area
  • The dealer spins the roulette wheel and place a ball on the wheel.
  • The ball will land on some number. Each number has an associated color.
  • The bets are checked if they match the outcome.
  • Bets are paid out differently depending on the chance of winning.
  • In this lab, we will write a class **Roulette** that can represent (ie simulate) spinning of the roulette wheel.

• Preparation:
• Create your ~/cs170/lab10/, and copy files by cutting and pasting these terminal commands:
  mkdir ~/cs170/lab11
  cp ~cs170001/share/lab11/*.java ~/cs170/lab11
  cd ~/cs170/lab11
• In this lab, we will use gedit and javac.
• You should see 6 Java files in your directory:
  • Roulette.java: this file will contain the definition of the roulette table (it will contain: (1) variables to hold information on the roulette table, and (2) methods that make the program behave like a roulette table)
  • Test1.java, Test2.java, Test3.java, Test4.java, and Test5.java: test programs to check if you have implemented various aspects of the roulette table correctly.

• Task 1: representing a roulette table
  • The function of a roulette table is to produce one of the following 38 possible outcomes:
    (0, Green)   (9, Red)   (18, Red)   (27, Red)   (36, Red)
    (1, Red)    (10, Black) (19, Red)   (28, Black) (00, Green)
    (2, Black)  (11, Black) (20, Black) (29, Black)
    (3, Red)    (12, Red)   (21, Red)   (30, Red)
    (4, Black)  (13, Black) (22, Black) (31, Black)
    (5, Red)    (14, Red)   (23, Red)   (32, Red)
    (6, Black)  (15, Black) (24, Black) (33, Black)
    (7, Red)    (16, Red)   (25, Red)   (34, Red)
    (8, Black)  (17, Black) (26, Black) (35, Black)
  • Notice that:
    • The "normal" outcomes are between 1 and 36, but there are 2 special outcomes: 0 and 00
    • Every outcome has a color associated with the outcome.
    • The "normal outcomes" (between 1 and 36) are either red or black
    • The "special outcomes" (0 and 00) are green.
  • We will need to store information to represent all these 38 outcomes
    We will use 2 arrays:
    String[] value;
    String[] color;
  • Take a look at these variables inside the Roulette.java program file:
    public class Roulette {
        public String[] value;  // Variables to represent
        public String[] color;  // the Roulette table
        public int outcome;

        /* ==-----------------------------*/
        Task 1: write the constructor
        ==-----------------------------*/
        public Roulette( ) {
        }
Note:

• The `value` and `color` variables are currently declared as `public` variables so that you can run the test program `Test1.java`
  • We will change the access specifier from `public` to `private` later in the lab.

• Write the constructor method `Roulette()` that must perform the following:
  • The constructor method `Roulette()` must create (with the `new` operator !) an array of 38 elements for `value`: to store the 38 values "0", "1", "2", ..., "35", "36", "00"
  • The constructor method `Roulette()` must create (with the `new` operator !) an array of 38 elements for `color`: to store the 38 values "G", "R", "B", "R", "B", ... ("G" means green, "R" means red and "B" means black)
    • The entries `value[i]` and `color[i]` store the value and the color for the one outcome
    • So make sure that the value and color of the outcome are correct (example, the outcome "1" has the color "Red", don't store the wrong color with that value !)
  • The constructor method `Roulette()` must store the roulette table information in the array (the roulette information is given above.)
• Testing the program. After writing the constructor method `Roulette`, you can test it with the `Test1.java` program. Compile and run the `Test1.java` file. You should see:

```java
Test1: constructor method in class Roulette

value[34] = 34 color[34] = R  value[35] = 35 color[35] = B
```

• **Task 2: define a `spin()` method that simulate a spin on the roulette wheel**
  • Recall that:
    • We have stored the 38 possible outcomes in the arrays `value` and `color`
    • Each one of the 38 entry of the arrays represents a outcome of a spin of the roulette wheel.
  • Task 2: Write the method `spin()` that records the outcome of a spin of the roulette wheel in the variable `outcome`:
public class Roulette {
    public String[] value;
    public String[] color;
    public int outcome;       // Stores the outcome of a spin
    ....

    /* ==============================================================
     * Task 2: write the spin() method
     * ============================================================== */
    public void spin() {
    }  
    .... (other methods omitted)
}

• Note:
  • The method spin() does not return any value.
  • Instead, the method spin() records (= updates) the outcome of a spin using the variable
    int outcome (we can retrieve the result from this variable !)
• The outcome can be represented by a random (integer) value between 0 and 37. We will use the
  value in the variable outcome to find the outcome information using the arrays value and
  color!
• Hints:
  • If you forgot how to generate a random number, take a look at this webpage: click here
  • You will need to multiply the random value and truncate it to an int using casting
• Testing the program:
  • After writing the method spin(), you can test it with Test2.java. Compile and run
    the Test2.java file. It should print out:

    Test2: spin method in class Roulette

    followed by a lot of numbers (each one should be between 0 and 37)
    You will see:

    Test was passed successfully !
    if the numbers are correct and:

    Illegal result of spin(): ...
    if you have a value that is < 0 or > 37
    If there is no value 37, the test program will say:

    The outcome 37 was not found; run test again

• Task 2b: changing instance variables from public to private
  • If your program has passed the Test2.java test, change the public access specifiers on the
    instance variables value, color, and outcome to private:
• Now, try re-compile the first 2 test programs, Test1.java and Test2.java:
  • You will get compilation errors, because the instance variables value, color and outcome can no longer be accessed from external classes.
  • Notice that before we made the change from public into private, the test programs Test1.java and Test2.java could access the variables value, color and outcome.
  • Therefore, we could make changes to these variables! In other words, we could ruin the correctness (for example, change the roulette table that will only spin the number 9!)
  • After changing the access specifiers from public into private, this "trick" is no longer possible !)

• **Task 3: write the value() method that returns the value of the spin**
  • Complete the value() method in the Roulette.java program and make the method return the string that represents the outcome of the value of the spin:
    ```java
    public class Roulette {
        public String[] value;  // Store the values of all outcomes
        public String[] color;
        public int outcome;     // represents the current outcome
        ...
        /* Task 3: write the value() method
            =========================================================================
            public String value() {
                return "";   // This return statement is wrong, write a correct one.
            }
        */
    }
    ```
  • Testing the program:
    • After writing the method value(), you can test it with Test3.java. Compile and run the Test3.java file.
    • The Test3.java program checks the number of times the roulette spin comes up with “13”.
      • The frequency should be approximately 26 times. If your program spins the number 13 more than 36 times or less than 16 times, check for errors.

• **Task 4: write the color() method that returns the color of the spin**
  • Complete the color() method in the Roulette.java program and make the method return the string that represents the outcome of the color of the spin:
    ```java
    public class Roulette {
        public String[] value;  // Store colors of all outcomes
    }
    ```
public int outcome;     // represents the current outcome
...

/* ==---------------------------------------------
 Task 4: write the color() method
=============================================== */
public String color() {
    return "";   // This return statement is wrong, write a correct one.
}
...

• Testing the program:
  • After writing the method color(), you can test it with Test4.java. Compile and run the Test4.java file.
  • The Test4.java program checks the number of times the roulette spin results in a red color ("R").
  • The frequency should be approximately 473 times.

• Task 5: write the toString() method
  • Take a look at the Test5.java program:

    public class Test5  {
        public static void main( String[] args ) {
            int i;
            int win=0, N=0;

            Roulette x = new Roulette( );

            System.out.println("Test5: toString method in class Roulette\n");

            N = 10;
            for ( i = 1; i <= N; i++ ) {
                x.spin();
                System.out.println( "x = " + x );
                // Converts a Roulette object x to a String !!!!
            }

            System.out.println();
            System.out.println("If you don't see '0 G' or '00 G', run again");
            System.out.println();
        }
    }

  • The Test5.java program will print a Roulette object as a String
    We will show you how to control the printing of objects that you define as a class.
  • Enter the follow toString() method into the Roulette.java program:

    /* ==---------------------------------------------
 Task 5: write the toString() method
=============================================== */
public String toString() {  // Write this toString method....
    return "Hello World !";
}

  • Compile and run Test5.java
• How does the Roulette objects get printed? (You should see 10 roulette objects printed, but the print out is "Hello World!" which is not very informative about a roulette object).
• Now write a `toString()` method inside `Roulette.java` that returns a String of the form:
  
  "value-of-the-spin color-of-the-spin"

• After you have written this method, compile and run `Test5.java`
• You should see an output like this:

  Test5: toString method in class Roulette
  x = 17 B
  x = 5 R
  x = 3 R
  x = 34 R
  x = 17 B
  x = 00 G
  x = 22 B
  x = 0 G
  x = 18 R
  x = 34 R

  If you don't see '0 G' or '00 G', run again

• **Turn in**
  • You must turn in your work by enter these terminal commands:
    
    cd ~/cs170/lab11

    /home/cs170XXX/turnin Roulette.java lab11