**Black-Box Test Plan (Simple, General CS3IOS Template):**

|  |  |
| --- | --- |
| **Test Case Scenario** | Guess the 3 or 4 or 5 digit number |
| **Test Technique Chosen** | Robust Boundary Value Analysis |
| **Reason for Test Technique Chosen** | I have chosen the robust boundary value analysis because it has outside range value and it cover all the test cases. |

|  |
| --- |
| **Code under Black-Box Testing (for marker's reference)** |
| import java.util.Random;  import java.util.Scanner;  class numbergame\_results{  private int digit;  private int chances;  private int randomnum;  private int usernumber;  private int counter=0;  public void setdigit(int digit){  this.digit=digit;  }  public void setChances(int chances){  this.chances=chances;  }  public int setRandomnumber(){  Random random = new Random();  if(digit ==3){  randomnum = random.nextInt(900)+100;  } else if (digit==4) {  randomnum = random.nextInt(9000)+1000;  } else if (digit==5) {  randomnum=random.nextInt(90000)+10000;  }  return randomnum;  }  public void setNumber(int usernumber){  this.usernumber=usernumber;  }  public void decision(){  for(int i=0;i<chances;i++){  if(Correct()==0){  CommonDigits();  System.out.println("Correct no of digits: "+counter);  counter=0;  Positions();  System.out.println("Correct Positions of digits: "+(counter));  counter=0;  System.out.println("Congratuations, you won");  break;  }else if(Correct()==1){  CommonDigits();  System.out.println("Correct no of digits: "+counter);  counter=0;  Positions();  System.out.println("Correct Positions of digits: "+counter);  counter=0;  break;  } else if (Correct() ==-1) {  CommonDigits();  System.out.println("Correct no of digits: "+counter);  counter=0;  Positions();  System.out.println("Corect Positions of digits: "+counter);  counter=0;  break;  }  }  }  public int Correct(){  if(usernumber>randomnum){  return -1;  }  else if(usernumber<randomnum){  return 1;  }else {  return 0;  }  }  public int CommonDigits(){  int temp=randomnum;  int temp1=usernumber;  int randomnum[] =new int[10];  int usernumber[]=new int[10];  while(temp>0){  randomnum[temp%10]++;  temp=temp/10;  }  while(temp1>0){  usernumber[temp1%10]++;  temp1=temp1/10;  }  for(int i=0;i<10;i++){  if(randomnum[i]>0 && usernumber[i]>0){  Counter();  }  }return 0;  }  public void Counter(){  counter++;  }  public void Positions(){  int num1,num2;  int temp= randomnum;  int temp1=usernumber;  while(temp!=0){  num1=temp%10;  temp=temp/10;  num2=temp1%10;  temp1=temp1/10;  if(num1==num2){  Counter();  }  }  }  }  abstract class numbergame\_Questions{  Scanner in=new Scanner(System.in)  ; public abstract void intro(String intro);  public abstract void start(String start);  public void digit(){System.out.println("Please enter which digit number you want to guess (3 or 4 or 5) ");}  public void three(){System.out.println("Enter a 3 digit number");}  public void chance(){System.out.println("In how many chances you can guess the number?");}  public void chances(){System.out.println("Your chances are completed");}  public void four(){System.out.println("Enter a 4 digit number");}  public void five(){System.out.println("Enter a 5 digit number");}  public void three\_incorrect(){System.out.println("Invalid input!, Enter a 3 digit number");}  public void four\_incorrect(){System.out.println("Invalid input!,Enter a 4 digit number");}  public void five\_incorrect(){System.out.println("Invalid input! ,Enter a 5 digit number");}  public void yes\_no(){System.out.println("Do you want to play again, Enter yes or no");}  public void wrongnumber(){System.out.println("Invalid input!, Enter 3 or 4 or 5 digit");}  public void thank(){System.out.println("Thank You");}  }  class numbergame\_decision extends numbergame\_Questions{  public void intro(String intro){  if(intro.equalsIgnoreCase("yes")){  System.out.println("Hi \n you must choose which digit number you want to guess? (eg: 3 or 4 or 5) \n Then you have to enter in how many chances you will guess the number. (eg: if you enter '5' then you have to guess the selected digit number in 5 chances) \n EX:3 digit number and 10 chances \n Computer generated random number (which you have to guess) is 523\n You have entered number as 678\n Then the program will give a hint as below \n Correct number of digit: 0 \n Correct positions of digits: 0\n anther number entered is 521\nCorrect number of digits: 2\nCorrect positons of digits: 2\n the above steps will be repeated until you guess the correct number or your chances get completed\nif your number is correct you will see a message as 'congratulation you won, do you want to play again(yes or no)' \nif your chances are completed you will see a message as ' your chances are completed, do you want to play again(yes or no)'" );  }  }  int chances,number;  numbergame\_results gamestart=new numbergame\_results();  public void start(String start){  if(start.equalsIgnoreCase("yes")){  String value;  do{  digit();  int digit=in.nextInt();  gamestart.setdigit(digit);  if(digit==3){  three\_digit();  }  else if(digit==4){  four\_digit();  }else if(digit==5){  five\_digit();  }else{  wrongnumber();  }  yes\_no();  value=in.next();  }  while (value.equalsIgnoreCase("yes"));  {  thank();  in.close();  }  }  else{  System.out.println("Closed");  }  }  public void three\_digit(){  chance();  chances=in.nextInt();  gamestart.setChances(chances);  int randomnum = gamestart.setRandomnumber();  for(int i=0;i<chances;i++){  three();  number=in.nextInt();  for (int j=i;j<=i;j++){  if(number >=100 && number <=999){  gamestart.setNumber(number);  gamestart.decision();  }  else{  three\_incorrect();  }  if(number == randomnum){  break;  }  }  if(number == randomnum){  break;  }  if(i == chances-1){  chances();  System.out.println("Correct number =" +randomnum);  }  }  }  public void four\_digit(){  chance();  chances=in.nextInt();  gamestart.setChances(chances);  int randomnum= gamestart.setRandomnumber();  for(int i=0;i<chances;i++){  four();  number=in.nextInt();  for(int j=i;j<=i;j++){  if(number > 999 && number <10000){  gamestart.setNumber(number);  gamestart.decision();  }  else{  four\_incorrect();  }  if(number == randomnum){  break;  }  }  if(number == randomnum){  break;  }  if(i == chances-1){  chances();  System.out.println("Correct number = "+ randomnum);  }  }  }  public void five\_digit(){  chance();  chances=in.nextInt();  gamestart.setChances(chances);  int randomnum=gamestart.setRandomnumber();  for(int i=0;i<chances;i++){  five();  number=in.nextInt();  for(int j=i;j<=i;j++){  if(number >9999 && number <100000){  gamestart.setNumber(number);  gamestart.decision();  }  else{  five\_incorrect();  }  if(number == randomnum){  break;  }  }  if(number == randomnum){  break;  }  if( i== chances -1){  chances();  System.out.println("Correct number = "+randomnum); }  }  }  }  public class numbergame\_App {  public static void main(String[] args){  Scanner kin=new Scanner(System.in);  System.out.println("Do you want to know about the game? (yes or no)");  numbergame\_Questions questions=new numbergame\_decision();  String intro=kin.next();  questions.intro(intro);  System.out.println("Let's start the game (yes or no) ");  String start=kin.next();  questions.start(start);  }  } |
| **Valid input range / set (for marker's reference)** |
| For 3 digit:  Range: 100<=number <=999  For 4 digit:  Range: 1000<= number <=9999  For 5 digit:  Range : 10000<=digit <= 99999 |

For 3 digit:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case Calculations (if appropriate)** | | | | | | |
| Extremes:   * Maximum Value = 999 * Minimum Value = 100   Near Extremes:   * Maximum Value -1 = 999-1 = 998 * Minimum Value +1= 999+1 =1000   Nominal:   * (Max-Min)/2 =(999-100)/2 = 449.5 =500   Outside range:   * Min-1 = 100-1 =99 * Max +1 =999+1 =1000   Guess number: 203 | | | | | | |
| **Test Case ID** | **Pre-Conditions** | **Test Step Details** | **Input(s) (UserNumber)** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| CS3IOS\_prj\_ID1\_3 | Enter integer number | Enter a 3 digit number | 99 | Invalid input!, Enter a 3 digit number | Invalid input!, Enter 3 digit number | Pass |
| CS3IOS\_prj\_ID2\_3 | Enter integer number | Enter a 3 digit number | 100 | Correct number of digits:1  Correct positions of digit:1 | Correct number of digits:1  Correct positions of digit:1 | Pass |
| CS3IOS\_prj\_ID3\_3 | Enter integer number | Enter a 3 digit number | 101 | Correct number of digits:1  Correct positions of digits:1 | Correct number of digits:1  Correct positions of digits:1 | Pass |
| CS3IOS\_prj\_ID4\_3 | Enter integer number | Enter a 3 digit number | 500 | Correct number of digits:1  Correct positions of digits:1 | Correct number of digits:1  Correct positions of digits:1 | Pass |
| CS3IOS\_prj\_ID5\_3 | Enter integer number | Enter a 3 digit number | 998 | Correct number of digits:0  Correct Positions of digits:0 | Correct number of digits:0  Correct positions of digits:0 | Pass |
| CS3IOS\_prj\_ID6\_3 | Enter integer number | Enter a 3 digit number | 999 | Correct number of digits:0  Correct Positions of digits:0 | Correct number of digits:0  Correct positions of digits:0 | Pass |
| CS3IOS\_prj\_ID7\_3 | Enter integer number | Enter a 3 digit number | 1000 | Invalid! Enter 3 digit number | Invalid ! Enter 3 digit number | Pass |

For 4 digit:

|  |
| --- |
| Test Case Calculations: |
| Extremes:   * Maximum Value = 9999 * Minimum Value =1000   Near Extremes:   * Maximum Value-1 =9999-1 = 9998 * Minimum Value +1 =9999+1=10000   Nominal:   * (Max-Min)/2=(9999-1000)/2 =4449.5=5000   Outside range:   * Min-1=1000-1=999 * Max+1=9999+1=10000 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Pre-Conditions | Test Step Details | Input(User Number) | Expected Results | Actual Results | Pass/ Fail |
| CS3IOS\_PRJ\_ID1\_4 | Enter integer number | Enter a 4 digit number | 999 | Invalid input ! Enter 3 digit number | Invalid input! Enter 3 digit number | Pass |
| CS3IOS\_PRJ\_ID2\_4 | Enter integer number | Enter a 4 digit number | 1000 | Correct number of digits:1  Correct number of positions:0 | Correct number of digits:1  Correct number of positions:0 | Pass |
| CS3IOS\_PRJ\_ID3\_4 | Enter integer number | Enter a 4 digit number | 1001 | Correct number of digits:1  Correct number of positions:0 | Correct number of digits:1  Correct number of positions:0 | Pass |
| CS3IOS\_PRJ\_ID4\_4 | Enter integer number | Enter a 4 digit number | 5000 | Correct number of digits:1  Correct number of positions:0 | Correct number of digits:1  Correct number of positions:0 | Pass |
| CS3IOS\_PRJ\_ID5\_4 | Enter integer number | Enter a 4 digit number | 9998 | Correct number of digits:0  Correct number of positions:0 | Correct number of digits:0  Correct number of positions:0 | Pass |
| CS3IOS\_PRJ\_ID6\_4 | Enter integer number | Enter a 4 digit number | 9999 | Correct number of digits:0  Correct number of Positions:0 | Correct number of digits:0  Correct number of positions:0 | Pass |
| CS3IOS\_PRJ\_ID7\_4 | Enter integer number | Enter a 4 digit number | 10000 | Invalid input! Enter 4 digit number | Invalid input! Enter 4 digit number | Pass |

For 5 digit:

|  |
| --- |
| Test Case Calculations |
| Test Cases : (6\*1)+1 =7  Extremes:   * Maximum value =99999 * Minimum value =10000   Near Extremes:   * Maximum Value-1 = 9999-1= 9998 * Minimum Value-1 = 9999+1=100000   Nominal:   * (Max-Min)/2 =(99999-100000)/2=44449.5=50000   Outside range:   * Min-1=10000-1=9999 * Max+1=99999+1=100000 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Pre\_conditions | Test step details | Input(User Number) | Expected Results | Actual Results | Pass / Fail |
| CS3IOS\_PRJ\_ID1\_5 | Enter integer number | Enter a 5 digit number | 9999 | Invalid input!, Enter 5 digit number | Invalid input!, Enter 5 digit number | Pass |
| CS3IOS\_PRJ\_ID2\_5 | Enter integer number | Enter a 5 digit number | 10000 | Correct number of digits:1  Correct number of positions:1 | Correct number of digits:1  Correct number of positions:1 | Pass |
| CS3IOS\_PRJ\_ID3\_5 | Enter integer number | Enter a 5 digit number | 10001 | Correct number of digits:1  Correct number of positions:1 | Correct number of digits:1  Correct number of positions:1 | Pass |
| CS3IOS\_PRJ\_ID4\_5 | Enter integer number | Enter a 5 digit number | 50000 | Correct number of digits:1  Correct number of Positions:1 | Correct number of digits:1  Correct number of positions:1 | Pass |
| CS3IOS\_PRJ\_ID5\_5 | Enter integer number | Enter a 5 digit number | 99998 | Correct number of digits:1  Correct number of Positions:1 | Correct number of digits:1  Correct number of positions:1 | Pass |
| CS3IOS\_PRJ\_ID6\_5 | Enter integer number | Enter a 5 digit number | 99999 | Correct number of digits:1  Correct number of Positions:1 | Correct number of digits:1  Correct number of Positions:1 | Pass |
| CS3IOS\_PRJ\_ID7\_5 | Enter integer number | Enter a 5 digit number | 100000 | Invalid input! Enter 5 digit number | Invalid input! Enter 5 digit number | Pass |

|  |
| --- |
| **Notes** |
| I chosen robust boundary value analysis, because all test cases are covered.  The part of the code which I chosen has one variable (i.e usernumber ).  *For 3 digit :*  *Range 100<=number<=999*  Test cases: CS3IOS\_PRJ\_ID1\_ 3 to CS3IOS \_PJ\_ID7\_3      *For 4 digit output:*  *Range 1000<= number <=9999*  *Testcases: CS3IOS\_PRJ\_ID1\_4 to CS3IOS\_PRJ\_ID7\_4*      *For 5 digit ouput:*  *Test Cases: CS3IOS\_PRJ\_ID1\_5 to Test Cases:CS3IOS\_PRJ\_ID7\_5*  *Range : 10000<= number <= 99999* |