Day3\_Assignment

Neha Khandave

Q.1.

|  |  |  |  |
| --- | --- | --- | --- |
| Sr.no | final | finally | finalize |
| 1. | keyword and access modifier | block in Java Exception Handling to execute the important code whether the exception occurs or no | method in Java which is used to perform clean up processing just before object is garbage collected. |
| 2 | classes, methods and variables. | related to the try and catch block in exception handling. | used with the objects. |
| 3. | (1) Once declared, final variable becomes constant and cannot be modified. (2) final method cannot be overridden by sub class. (3) final class cannot be inherited. | (1) finally block runs the important code even if exception occurs or not. (2) finally block cleans up all the resources used in try block | finalize method performs the cleaning activities with respect to the object before its destruction |
| 4 | //declaring final variable  reassigning value to age variable gives compile time error  **public** **class** FinalTest {  **final** **int** age = 58;  **void** display() {      age = 55;      }  **public** **static** **void** main(String[] args) {    FinalTest obj = **new** FinalTest();   obj.display();   // gives compile time error   }  } | **public** **static** **void** main(String args[]){  **try** {          System.out.println("Inside try block");  **int** data=15/0;          //  code throws divide by zero exception  System.out.println(data);        }        // handles the Arithmetic Exception / Divide by zero exception  **catch** (ArithmeticException e){          System.out.println("Exception handled");          System.out.println(e);        }        // executes regardless of exception occurred or not  **finally** {          System.out.println("finally block is always executed");        }        System.out.println("rest of the code...");        }      } | **public** **class** FinalizeExample {  **public** **static** **void** main(String[] args)      {          FinalizeExample obj = **new** FinalizeExample();          System.out.println("Hashcode is: " + obj.hashCode());  // printing the hashcode          obj = **null**;      System.gc();     // calling the garbage collector using gc()       System.out.println("End of the garbage collection");      }  **protected** **void** finalize()        // defining the finalize method      {          System.out.println("Called the finalize() method");      } |

Q.2.

**import** java.util.Scanner;

**class** RideCab{

**int** fare;

**int** d;

**public** RideCab(){

fare = 50;

}

**public** RideCab(**int** amt){

fare = amt;

}

}

**class** Cab{

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

System.***out***.println("Enter the cabdistance frm u ");

**int** cd;

cd =sc.nextInt();

**if**(cd >5) {

RideCab ob = **new** RideCab(50 + 10\* (cd-5));//base chrge will be 8-5=3,amt=80

System.***out***.println("Enter the distance travelled");

ob.d =sc.nextInt();

ob.fare += 10 \*ob.d;//80+60=140

System.***out***.println("Total fare = " + ob.fare);

}**else** {

RideCab ob = **new** RideCab();

System.***out***.println("Enter the distance travelled");

ob.d =sc.nextInt();

ob.fare += 10 \*ob.d;

System.***out***.println("Total fare = " + ob.fare);

}

}

}

Output:

Enter the cabdistance frm u

4

Enter the distance travelled

6

Total fare = 110

Enter the cabdistance frm u

4

Enter the distance travelled

4

Total fare = 90