| Nama | Adidarma Putra Harjuna |
| --- | --- |
| NRP | 5221600027 |
| Topik | Classes 2 |

Referensi: https://drive.google.com/file/d/1VMEDwkhcJf4xhjwFBwbpxOzpIfslFAII/view?usp=drive\_web&authuser=0

Pelajari materi **chapter 7-8** (terkait class dan constructor) buku referensi dan Kerjakan latihan berikut ini dengan mengikuti petunjuk yang diberikan.

| 1 | Buat kelas *Fraction* dengan tiga three private data fields untuk bilangan bulat, pembilang, dan penyebut. Tambahkan pula constant static public field untuk menyimpan symbol pemisah pembilang dan penyebut ketika menampilkan Fraction — tanda ‘/’ . Buat tiga public member functions dalam kelas, sebagai berikut:   * An *enterFractionValue*() function that prompts the user to enter values for the Fraction. Do not allow the user to enter a value of 0 for the denominator of any Fraction; continue to prompt the user for a denominator value until a valid one is entered. * A *reduceFraction*()function that reduces a Fraction to proper form. For example, a Fraction with the value 0 2/6 would be reduced to 0 1/3 and a Fraction with the value 4 18/4 would be reduced to 8 1/2. * A *displayFraction*()function that displays the Fraction whole number, numerator, slash, and denominator.   Tambahkan fungsi lain yang dibutuhkan kalian. Buat fungsi *main*() yang membuat object *Fraction*, dan meminta user untuk memasukkan nilai pada object Fraction tersebut. Setiap fraction yang dimasukkan, tampilkan pecahan, sederhanakan pecahan dan tampilkan pecahan lagi.  Hasil:  #include <iostream>  using namespace std;  class fraction  {  private:  int bilbulat;  int pembilang;  int penyebut;  public:  int enterfractionValue();  void displayfraction();  void reducefraction();  int getpembilang()  {  return pembilang;  }  int getbilbulat()  {  return pembilang;  }  };  int fraction::enterfractionValue()  {  cout << "\nMasukkan Bilangan bulat : ";  cin >> bilbulat;  cout << "\nMasukkan Pembilang : ";  cin >> pembilang;  cout << "\nMasukkan Penyebut : ";  cin >> penyebut;  while (penyebut == 0)  {  cout << "Penyebut Invalid";  cout << "Mohon Masukkan Ulang";  }  }  void fraction::displayfraction()  {  cout << "\nreduced fraction : "  << "" << bilbulat << "" << pembilang << "/" << penyebut;  }  void fraction::reducefraction()  {  int complete = bilbulat \* penyebut + pembilang;  bilbulat = complete / penyebut;  pembilang = complete % penyebut;  if ((pembilang % 2 == 0) && (penyebut % 2 == 0))  {  pembilang = pembilang / 2;  penyebut = penyebut / 2;  }  }  int main()  {  int cal = 0;  fraction newfraction;  while (cal == 0)  {  newfraction.enterfractionValue();  if ((newfraction.getbilbulat() == 0) && (newfraction.getpembilang() == 0))  cal = 1;  break;  }  newfraction.reducefraction();  newfraction.displayfraction();  newfraction.reducefraction();  system("pause");  return 0;  }    Kesimpulan:  (berikan kesimpulan disini) |
| --- | --- |
| 2 | Modifikasi program no 1 diatas sebagai berikut::  Add two constructors to the class.  The first accepts two integer values representing the numerator and denominator. If a single integer is passed to the constructor, use it as the numerator, and use a default value of 1 for the denominator. If no values are passed to the constructor, use a default value of 0 for the numerator and 1 for the denominator. When any Fraction is constructed with a 0 argument for the denominator, force the denominator value to 1.  The second constructor requires three arguments: a whole number portion for a Fraction, a numerator, and a denominator. This constructor executes when any Fraction object is instantiated using three integer arguments. As with the other constructor, when any Fraction is constructed with a 0 argument for the denominator, force the denominator value to 1.  Whenever a Fraction object is constructed, automatically reduce the Fraction to the proper format. For example, a Fraction created using 0, 2, and 4 as arguments should be reduced to 1/2, and a Fraction created as 3 10/2 should be reduced to 8 0/1.  Write a main() function that declares several Fraction objects, and confirm that the class works correctly.  Hasil:  (capture source code disini)  (capture output/tampilan program disini)  Kesimpulan:  (berikan kesimpulan disini) |