TUGAS PRAKTEK OOP DASAR

Menggunakan Bahasa Pemrograman Java Dan C++

Dosen Pengampu :

Bapak Asmunin, S.Kom., M.Kom.

Disusun oleh :

Eva Fitria Novianti Putri

22091397068

Pemrograman Berorientasi Objek (2022B)

Prodi D4 Manajemen Informatika

Fakultas Vokasi

Universitas Negeri Surabaya

2023

Praktek Class Menggunakan Bahasa Pemroggaman Java Dan C++

Class:

1. Circle
2. Date
3. Point
4. Time
5. Account

JAVA

1. Circle

- **Source Code**

  - Circle.java

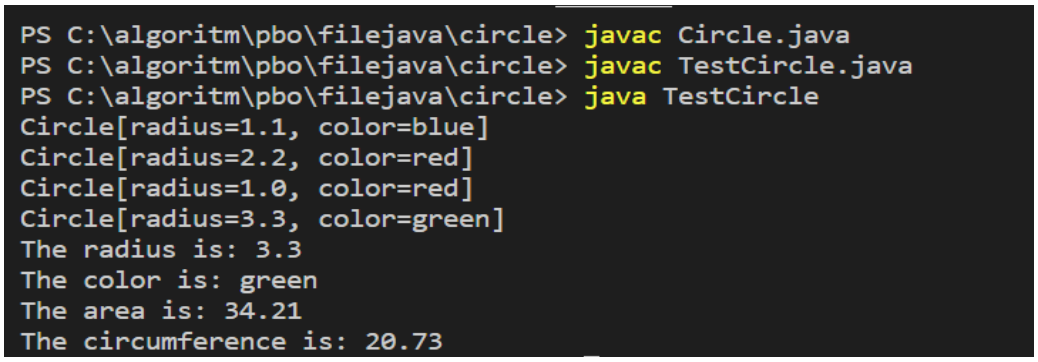
```java
/* The Circle class models a circle with a radius and color. */
public class Circle {    // Save as "Circle.java"
   // The public constants
   public static final double DEFAULT_RADIUS = 1.0;
   public static final String DEFAULT_COLOR  = "red";

   // The private instance variables
   private double radius;
   private String color;

   // The (overloaded) constructors
   /** Constructs a Circle with default radius and color */
   public Circle() {                    // 1st (default) Constructor
      this.radius = DEFAULT_RADIUS;
      this.color  = DEFAULT_COLOR;
   }
   /** Constructs a Circle with the given radius and default color */
   public Circle(double radius) {      // 2nd Constructor
      this.radius = radius;
      this.color = DEFAULT_COLOR;
   }
   /** Constructs a Circle with the given radius and color */
   public Circle(double radius, String color) { // 3rd Constructor
      this.radius = radius;
      this.color = color;
   }

   /** Returns the radius - the public getter for private variable
radius. */
   public double getRadius() {
      return this.radius;
   }
   /** Sets the radius - the public setter for private variable radius
*/
   public void setRadius(double radius) {
      this.radius = radius;
   }
   /** Returns the color - the public getter for private variable color
*/
   public String getColor() {
      return this.color;
   }
   /** Sets the color - the public setter for private variable color */
   public void setColor(String color) {
      this.color = color;
   }

   /** Returns a self-descriptive string for this Circle instance */
   public String toString() {
      return "Circle[radius=" + radius + ", color=" + color + "]";
   }

   /** Returns the area of this Circle */
   public double getArea() {
      return radius * radius * Math.PI;
   }

   /** Returns the circumference of this Circle */
   public double getCircumference() {
      return 2.0 * radius * Math.PI;
   }
}
```

- **TestCircle.java**

```java
/* A Test Driver for the Circle class */
public class TestCircle {
   public static void main(String[] args) {
      // Test all constructors and toString()
      Circle c1 = new Circle(1.1, "blue");
      System.out.println(c1);  // implicitly run toString()
```

```
        //Circle[radius=1.1, color=blue]
        Circle c2 = new Circle(2.2);
        System.out.println(c2);
        //Circle[radius=2.2, color=red]
        Circle c3 = new Circle();
        System.out.println(c3);
        //Circle[radius=1.0, color=red]

        // Test Setters and Getters
        c1.setRadius(3.3);
        c1.setColor("green");
        System.out.println(c1);  //use toString() to inspect the modified
instance
        //Circle[radius=3.3, color=green]
        System.out.println("The radius is: " + c1.getRadius());
        //The radius is: 3.3
        System.out.println("The color is: " + c1.getColor());
        //The color is: green

        // Test getArea() and getCircumference()
        System.out.printf("The area is: %.2f%n", c1.getArea());
        //The area is: 34.21
        System.out.printf("The circumference is: %.2f%n",
c1.getCircumference());
        //The circumference is: 20.73
    }
}
```

- **Output**



```
PS C:\algoritm\pbo\filejava\circle> javac Circle.java
PS C:\algoritm\pbo\filejava\circle> javac TestCircle.java
PS C:\algoritm\pbo\filejava\circle> java TestCircle
Circle[radius=1.1, color=blue]
Circle[radius=2.2, color=red]
Circle[radius=1.0, color=red]
Circle[radius=3.3, color=green]
The radius is: 3.3
The color is: green
The area is: 34.21
The circumference is: 20.73
```

2. **Date**

- **Source Code**

  - **Date.java**

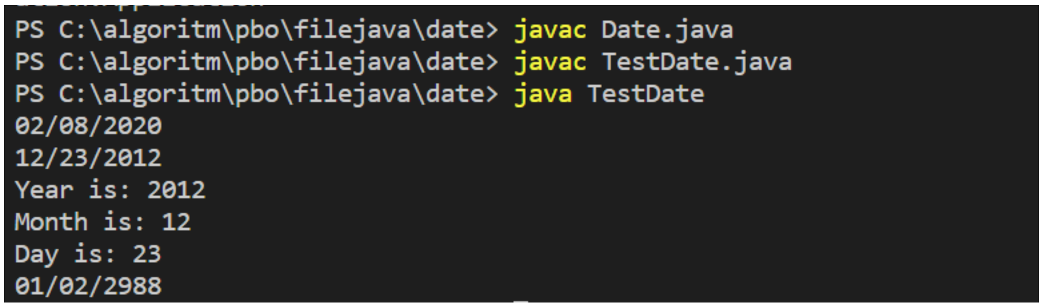
```
/* The Date class models a calendar date with day, month and year.
 * This class does not perform input validation for day, month and
year. */
public class Date {
   // The private instance variables
   private int year, month, day;

   /** Constructs a Date instance with the given year, month and day.
No input validation */
   public Date(int year, int month, int day) {
      this.year = year;
      this.month = month;
      this.day = day;
   }

   // The public getters/setters for the private variables
   /** Returns the year */
   public int getYear() {
      return this.year;
   }
   /** Returns the month */
   public int getMonth() {
      return this.month;
   }
   /** Returns the day */
   public int getDay() {
      return this.day;
   }
   /** Sets the year. No input validation */
   public void setYear(int year) {
      this.year = year;
   }
   /** Sets the month. No input validation */
   public void setMonth(int month) {
      this.month = month;
   }
   /** Sets the day. No input validation */
   public void setDay(int day) {
      this.day = day;
   }

   /* Returns a descriptive String in the form "MM/DD/YYYY" with
leading zero */
   public String toString() {
```

```java
        // Use built-in function String.format() to form a formatted
String
        return String.format("%02d/%02d/%4d", month, day, year);
              // Specifier "0" to print leading zeros
   }

   /** Sets year, month and day. No input validation */
   public void setDate(int year, int month, int day) {
      this.year = year;
      this.month = month;
      this.day = day;
   }
}
```

- **TestDate.java**

```java
/* A Test Driver for the Date class. */
public class TestDate {
   public static void main(String[] args) {
      // Test constructor and toString()
      Date d1 = new Date(2020, 2, 8);
      System.out.println(d1);  // toString()
      //02/08/2020

      // Test Setters and Getters
      d1.setYear(2012);
      d1.setMonth(12);
      d1.setDay(23);
      System.out.println(d1);

      //12/23/2012
      System.out.println("Year is: " + d1.getYear());
      //Year is: 2012
      System.out.println("Month is: " + d1.getMonth());
      //Month is: 12
      System.out.println("Day is: " + d1.getDay());
      //Day is: 23

      // Test setDate()
      d1.setDate(2988, 1, 2);
      System.out.println(d1);
      //01/02/2988
   }
}
```

- **Output**

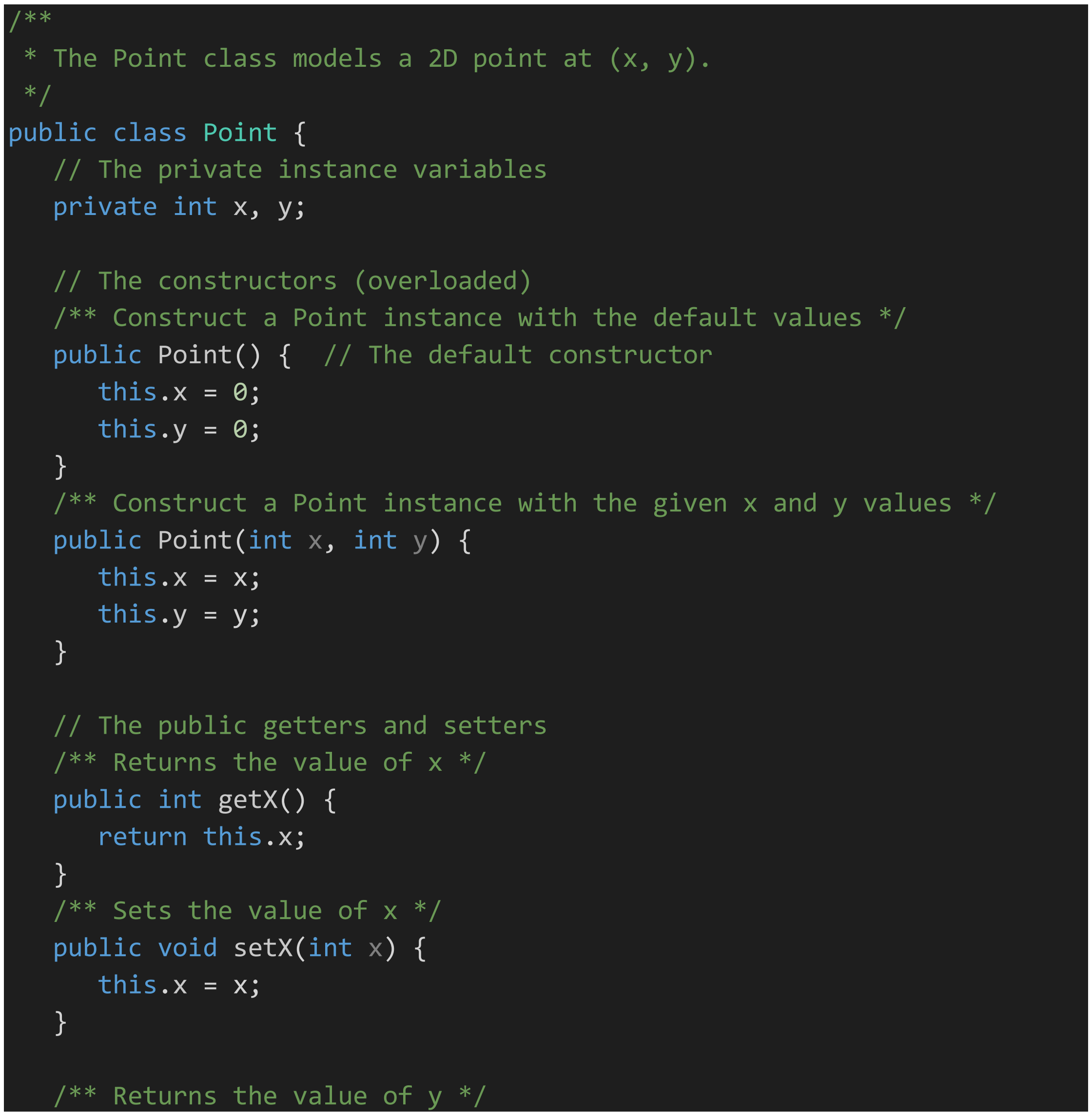


```
PS C:\algoritm\pbo\filejava\date> javac Date.java
PS C:\algoritm\pbo\filejava\date> javac TestDate.java
PS C:\algoritm\pbo\filejava\date> java TestDate
02/08/2020
12/23/2012
Year is: 2012
Month is: 12
Day is: 23
01/02/2988
```

3. **Point**

- **Source Code**

  - **Point.java**

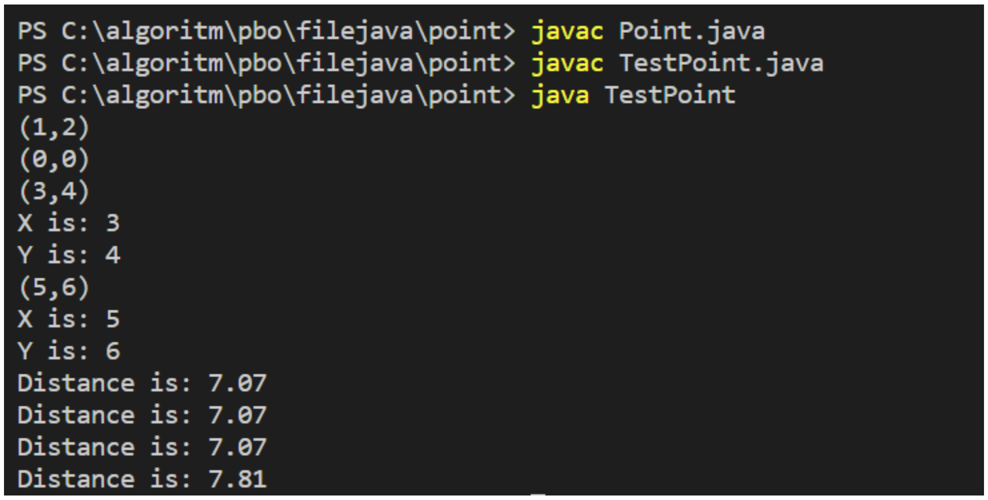


```java
/**
 * The Point class models a 2D point at (x, y).
 */
public class Point {
   // The private instance variables
   private int x, y;

   // The constructors (overloaded)
   /** Construct a Point instance with the default values */
   public Point() {  // The default constructor
      this.x = 0;
      this.y = 0;
   }
   /** Construct a Point instance with the given x and y values */
   public Point(int x, int y) {
      this.x = x;
      this.y = y;
   }

   // The public getters and setters
   /** Returns the value of x */
   public int getX() {
      return this.x;
   }
   /** Sets the value of x */
   public void setX(int x) {
      this.x = x;
   }

   /** Returns the value of y */
   public int getY() {
      return this.y;
   }
   /** Sets the value of y */
   public void setY(int y) {
      this.y = y;
   }

   /** Returns a self-descriptive string in the form of "(x,y)" */
   public String toString() {
      return "(" + this.x + "," + this.y + ")";
   }

   /** Returns a 2-element int array containing x and y */
   public int[] getXY() {
      int[] results = new int[2];
      results[0] = this.x;
      results[1] = this.y;
      return results;
   }

   /** Sets both x and y */
   public void setXY(int x, int y) {
      this.x = x;
      this.y = y;
   }

   /** Return the distance from this instance to the given point at
(x,y). Invoke via p1.distance(1,2) */
   public double distance(int x, int y) {
      int xDiff = this.x - x;
      int yDiff = this.y - y;
      return Math.sqrt(xDiff*xDiff + yDiff*yDiff);
   }
   /** Returns the distance from this instance to the given Point
instance. Invoke via p1.distance(p2) */
   public double distance(Point another) {
      int xDiff = this.x - another.x;
      int yDiff = this.y - another.y;
      return Math.sqrt(xDiff*xDiff + yDiff*yDiff);
   }
   /** Returns the distance from this instance to (0,0). Invoke via
p1.distance() */
   public double distance() {
      return Math.sqrt(this.x*this.x + this.y*this.y);
   }
}
```

- **TestPoint.java**

```
/**
 * A Test Driver for the Point class.
 */
public class TestPoint {
   public static void main(String[] args) {
      // Test constructors and toString()
      Point p1 = new Point(1, 2);
      System.out.println(p1);  // toString()
      //(1,2)
      Point p2 = new Point();  // default constructor
      System.out.println(p2);
      //(0,0)

      // Test Setters and Getters
      p1.setX(3);
      p1.setY(4);
      System.out.println(p1);  // run toString() to inspect the
modified instance
      //(3,4)
      System.out.println("X is: " + p1.getX());
      //X is: 3
      System.out.println("Y is: " + p1.getY());
      //Y is: 4

      // Test setXY() and getXY()
      p1.setXY(5, 6);
      System.out.println(p1);  // toString()
      //(5,6)
      System.out.println("X is: " + p1.getXY()[0]);
      //X is: 5
      System.out.println("Y is: " + p1.getXY()[1]);
      //Y is: 6

      // Test the 3 overloaded versions of distance()
      p2.setXY(10, 11);
      System.out.printf("Distance is: %.2f%n", p1.distance(10, 11));
      //Distance is: 7.07
      System.out.printf("Distance is: %.2f%n", p1.distance(p2));
      //Distance is: 7.07
      System.out.printf("Distance is: %.2f%n", p2.distance(p1));
      //Distance is: 7.07
      System.out.printf("Distance is: %.2f%n", p1.distance());
      //Distance is: 7.81
   }
}
```

- **Output**



```
PS C:\algoritm\pbo\filejava\point> javac Point.java
PS C:\algoritm\pbo\filejava\point> javac TestPoint.java
PS C:\algoritm\pbo\filejava\point> java TestPoint
(1,2)
(0,0)
(3,4)
X is: 3
Y is: 4
(5,6)
X is: 5
Y is: 6
Distance is: 7.07
Distance is: 7.07
Distance is: 7.07
Distance is: 7.81
```

4. Time

- **Source Code**

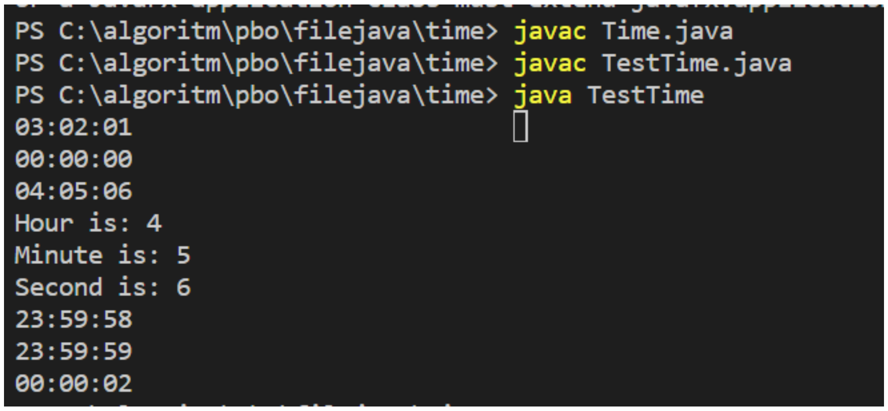
  - **Time.java**

```
/* The Time class models a time instance with second, minute and hour.
 * This class does not perform input validation for second, minute and
hour. */
public class Time {
   // The private instance variables
   private int second, minute, hour;

   // The constructors (overloaded)
   /** Constructs a Time instance with the given second, minute and
hour. No input validation */
   public Time(int second, int minute, int hour) {
      this.second = second;
      this.minute = minute;
      this.hour = hour;
   }
   /** Constructs a Time instance with the default values */
   public Time() {  // the default constructor
      this.second = 0;
      this.minute = 0;
      this.hour = 0;
   }

   // The public getters/setters for the private variables.
   /** Returns the second */
   public int getSecond() {
      return this.second;
   }
   /** Returns the minute */
   public int getMinute() {
      return this.minute;
   }
   /** Returns the hour */
   public int getHour() {
      return this.hour;
   }
   /** Sets the second. No input validation */
   public void setSecond(int second) {
      this.second = second;
   }
   /** Sets the minute. No input validation */
   public void setMinute(int minute) {
      this.minute = minute;
   }
   /** Sets the hour. No input validation */
   public void setHour(int hour) {
      this.hour = hour;
   }

   /** Returns a self-descriptive string in the form of  "hh:mm:ss"
with leading zeros */
   public String toString() {
        // Use built-in function String.format() to form a formatted
String
        return String.format("%02d:%02d:%02d", hour, minute, second);
            // Specifier "0" to print leading zeros, if available.
   }

   /** Sets second, minute and hour to the given values */
   public void setTime(int second, int minute, int hour) {
      // No input validation
      this.second = second;
      this.minute = minute;
      this.hour = hour;
   }
   /** Advances this Time instance by one second, and returns this
instance to support chaining */
   public Time nextSecond() {
      ++second;
      if (second >= 60) {
         second = 0;
         ++minute;
         if (minute >= 60) {
            minute = 0;
            ++hour;
            if (hour >= 24) {
               hour = 0;
            }
         }
      }
      return this;  // Return "this" instance, to support chaining
operations
                    // e.g., t1.nextSecond().nextSecond()
   }
}
```

- **TestTime.java**

```
/* A Test Driver for the Time class */
public class TestTime {
   public static void main(String[] args) {
      // Test Constructors and toString()
      Time t1 = new Time(1, 2, 3);
      System.out.println(t1);  // toString()
      //03:02:01
      Time t2 = new Time();    // The default constructor
      System.out.println(t2);
      //00:00:00

      // Test Setters and Getters
      t1.setHour(4);
      t1.setMinute(5);
      t1.setSecond(6);
      System.out.println(t1);  // run toString() to inspect the
modified instance
      //04:05:06
      System.out.println("Hour is: " + t1.getHour());
      //Hour is: 4
      System.out.println("Minute is: " + t1.getMinute());
      //Minute is: 5
      System.out.println("Second is: " + t1.getSecond());
      //Second is: 6

      // Test setTime()
      t1.setTime(58, 59, 23);
      System.out.println(t1);
      //23:59:58

      // Test nextSecond() and chaining
      System.out.println(t1.nextSecond()); // Return an instance of
Time. Invoke Time's toString()
      //23:59:59
      System.out.println(t1.nextSecond().nextSecond().nextSecond());  /
/ chaining
      //00:00:02
   }
}
```

- **Output**



```
PS C:\algoritm\pbo\filejava\time> javac Time.java
PS C:\algoritm\pbo\filejava\time> javac TestTime.java
PS C:\algoritm\pbo\filejava\time> java TestTime
03:02:01
00:00:00
04:05:06
Hour is: 4
Minute is: 5
Second is: 6
23:59:58
23:59:59
00:00:02
```

5. **Account**

- **Source Code**

  - **Account.java**

```
/**
 * The Account class models a bank account with a balance.
 */
public class Account {
   // The private instance variables
   private int number;
   private double balance;

   // The constructors (overloaded)
   /** Constructs an Account instance with the given number and initial
balance of 0 */
```

```
   public Account(int number) {
      this.number = number;
      this.balance = 0.0;  // "this." is optional
   }
   /** Constructs an Account instance with the given number and initial
balance */
   public Account(int number, double balance) {
      this.number = number;
      this.balance = balance;
   }

   // The public getters/setters for the private instance variables.
   // No setter for number because it is not designed to be changed.
   // No setter for balance as it is changed via credit() and debit()
   /** Returns the number */
   public int getNumber() {
      return this.number;  // "this." is optional
   }
   /** Returns the balance */
   public double getBalance() {
      return this.balance;  // "this." is optional
   }

   /** Returns a string description of this instance */
   public String toString() {
      // Use built-in function System.format() to form a formatted
String
      return String.format("Account[number=%d,balance=$%.2f]", number,
balance);
   }

   /** Add the given amount to the balance */
   public void credit(double amount) {
      balance += amount;
   }

   /** Subtract the given amount from balance, if balance >= amount */
   public void debit(double amount) {
      if (balance >= amount) {
         balance -= amount;
      } else {
         System.out.println("amount exceeded");
      }
   }
```

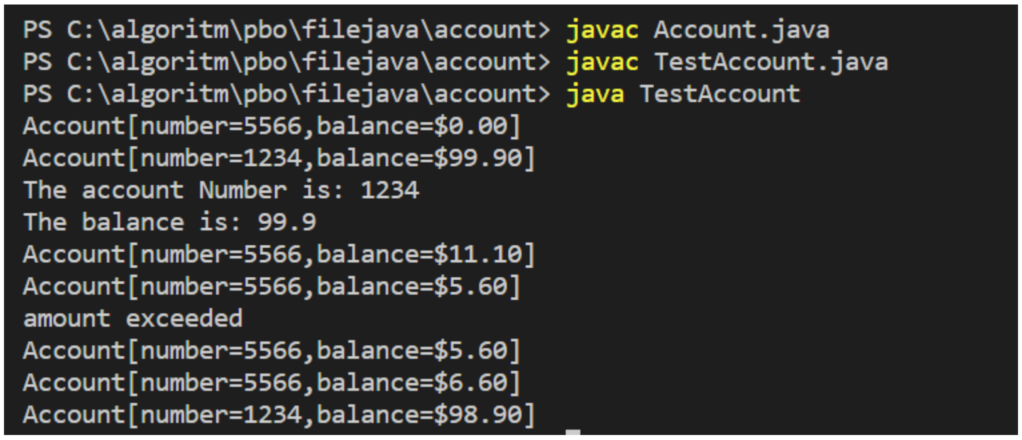
```java
   /** Transfer the given amount to Account another, if balance >= 
amount */
   public void transferTo(double amount, Account another) {
      if (balance >= amount) {
         this.balance -= amount;
         another.balance += amount;
      } else {
         System.out.println("amount exceeded");
      }
   }
}
```

- **TestAccount.java**

```java
/**
 * A Test Driver for the Account class.
 */
public class TestAccount {
   public static void main(String[] args) {
      // Test Constructors and toString()
      Account a1 = new Account(5566);
      System.out.println(a1);
      //Account[number=5566,balance=$0.00]
      Account a2 = new Account(1234, 99.9);
      System.out.println(a2);
      //Account[number=1234,balance=$99.90]

      // Test getters
      System.out.println("The account Number is: " + a2.getNumber());
      //The account Number is: 1234
      System.out.println("The balance is: " + a2.getBalance());
      //The balance is: 99.9

      // Test credit(), debit() and transferTo()
      a1.credit(11.1);
      System.out.println(a1);
      //Account[number=5566,balance=$11.10]
      a1.debit(5.5);
      System.out.println(a1);
      //Account[number=5566,balance=$5.60]
      a1.debit(500);   // Test debit() error
      //amount exceeded
      System.out.println(a1);
      //Account[number=5566,balance=$5.60]

      a2.transferTo(1.0, a1);
```

```
        System.out.println(a1);
        //Account[number=5566,balance=$6.60]
        System.out.println(a2);
        //Account[number=1234,balance=$98.90]
    }
}
```

- **Output**



```
PS C:\algoritm\pbo\filejava\account> javac Account.java
PS C:\algoritm\pbo\filejava\account> javac TestAccount.java
PS C:\algoritm\pbo\filejava\account> java TestAccount
Account[number=5566,balance=$0.00]
Account[number=1234,balance=$99.90]
The account Number is: 1234
The balance is: 99.9
Account[number=5566,balance=$11.10]
Account[number=5566,balance=$5.60]
amount exceeded
Account[number=5566,balance=$5.60]
Account[number=5566,balance=$6.60]
Account[number=1234,balance=$98.90]
```

C++

1. Circle

- **Source Code**

  - **Circle.h**

```cpp
/*The Circle class Header (Circle.h)*/
//Library yang digunakan
#include <string>
using namespace std;

//Mendeklarasikan Circle class
class Circle{
private: //Hanya dapat diakses oleh anggota class ini
    //Private data anggota (variabel)
    double radius;
    string color;

public: //Dapat diakses oleh semua orang
    //Mendeklarasikan protoype fungsi anggota
    //Pembuat dengan nilai default
    Circle(double radius = 1.0, string color = "red");

    // Public getters & setters for private data members
    double getRadius() const;
    void setRadius(double radius);
    string getColor() const;
    void setColor(string color);

    //Public member Fungtion
    double getArea() const;
};
```

  - **Circle.cpp**

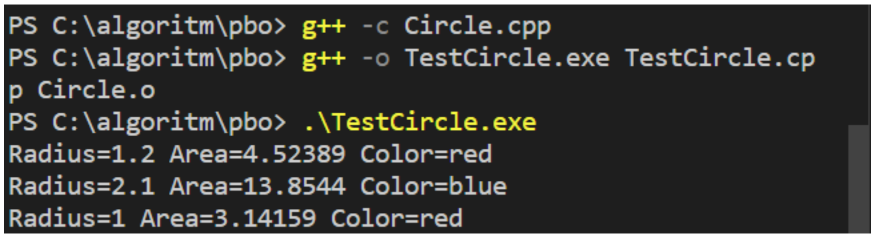
```cpp
/* The Circle class Implementation (Circle.cpp)*/
//Library yang digunakan
#include "Circle.h"

//Constructor
//Nilai default hanya boleh ditentukan dalam deklarasi, tidak dapat
diulangi dalam definisi
Circle::Circle(double r, string c){
    radius = r;
    color = c;
}
```

```cpp

//Public getter untuk radius
double Circle::getRadius() const{
    return radius;
}

//Public setter untuk radius
void Circle::setRadius(double r){
    radius = r;
}

//Public getter untuk color
string Circle::getColor() const{
    return color;
}

//Public setter untuk color
void Circle::setColor(string c){
    color = c;
}

//A public member fungtion
double Circle::getArea() const{
    return radius*radius*3.14159265;
}
```

- **TestCircle.cpp**

```cpp
/* A test driver for the Circle class (TestCircle.cpp)*/
#include <iostream>
#include "Circle.h"

using namespace std;

int main(){
    //Membuat sebuah instance dari Circle c1
    Circle c1(1.2, "red");
    cout<< "Radius=" << c1.getRadius()
        << " Area=" << c1.getArea()
        << " Color=" << c1.getColor() <<endl;

    c1.setRadius(2.1); //Mengubah radius dan color c1
    c1.setColor("blue");
    cout<< "Radius=" << c1.getRadius()
        << " Area=" << c1.getArea()
        << " Color=" << c1.getColor() <<endl;
    //Membuat instance lain menggunakan konstuktor default
    Circle c2;
    cout<< "Radius=" << c2.getRadius()
        << " Area=" << c2.getArea()
        << " Color=" << c2.getColor() <<endl;

    return 0;
}
```

- **Output**



```
PS C:\algoritm\pbo> g++ -c Circle.cpp
PS C:\algoritm\pbo> g++ -o TestCircle.exe TestCircle.cp
p Circle.o
PS C:\algoritm\pbo> .\TestCircle.exe
Radius=1.2 Area=4.52389 Color=red
Radius=2.1 Area=13.8544 Color=blue
Radius=1 Area=3.14159 Color=red
```

2. **Date**

- **Source Code**

  - **Date.h**

```cpp
#ifndef DATE_H
#define DATE_H

#include <string>

class Date {
private:
    int year, month, day;

public:
    Date(int year = 1, int month = 1, int day = 1);
    int getYear() const;
    int getMonth() const;
    int getDay() const;
    void setYear(int year);
    void setMonth(int month);
    void setDay(int day);
    std::string toString() const;
    void setDate(int year, int month, int day);
};

#endif  // DATE_H
```

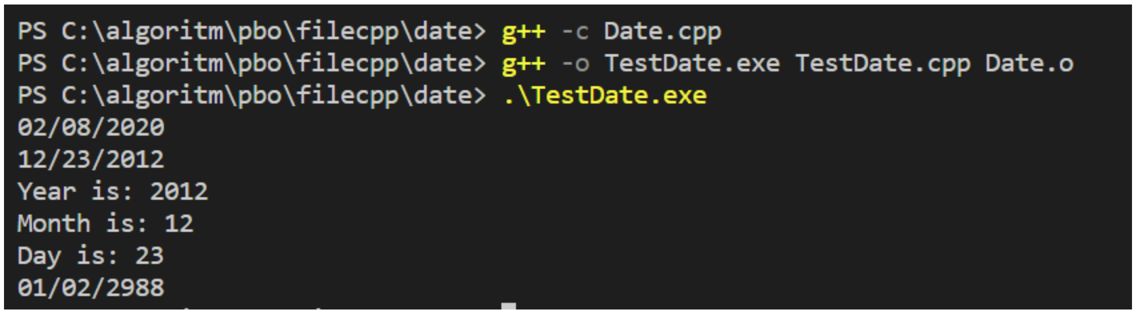
- **Date.cpp**

```cpp
#include "Date.h"
#include <iostream>
#include <iomanip>

Date::Date(int year, int month, int day)
    : year(year), month(month), day(day) {}

int Date::getYear() const {
    return year;
}

int Date::getMonth() const {
    return month;
}

int Date::getDay() const {
    return day;
}

void Date::setYear(int year) {
    this->year = year;
}

void Date::setMonth(int month) {
    this->month = month;
}

void Date::setDay(int day) {
    this->day = day;
}

void Date::setDate(int year, int month, int day) {
    this->year = year;
    this->month = month;
    this->day = day;
}

std::string Date::toString() const {
    std::ostringstream oss;
    oss << std::setfill('0') << std::setw(2) << month << "/"
        << std::setw(2) << day << "/" << std::setw(4) << year;
    return oss.str();
}
```

- **TestDate.cpp**

```cpp
#include <iostream>
#include "Date.h"

int main() {
    // Test constructor and toString()
    Date d1(2020, 2, 8);
    std::cout << d1.toString() << std::endl;  // toString()
    // 02/08/2020

    // Test Setters and Getters
    d1.setYear(2012);
    d1.setMonth(12);
    d1.setDay(23);
    std::cout << d1.toString() << std::endl;
    // 12/23/2012
    std::cout << "Year is: " << d1.getYear() << std::endl;
    // Year is: 2012
    std::cout << "Month is: " << d1.getMonth() << std::endl;
    // Month is: 12
    std::cout << "Day is: " << d1.getDay() << std::endl;
    // Day is: 23

    // Test setDate()
    d1.setDate(2988, 1, 2);
    std::cout << d1.toString() << std::endl;
    // 01/02/2988

    return 0;
}
```

- **Output**



```
PS C:\algoritm\pbo\filecpp\date> g++ -c Date.cpp
PS C:\algoritm\pbo\filecpp\date> g++ -o TestDate.exe TestDate.cpp Date.o
PS C:\algoritm\pbo\filecpp\date> .\TestDate.exe
02/08/2020
12/23/2012
Year is: 2012
Month is: 12
Day is: 23
01/02/2988
```

3. **Point**

- **Source Code**

- **Point.h**

```
/* The Point class Header (Point.h) */
#ifndef POINT_H
#define POINT_H

// Point class declaration
class Point {
private:
   // private data members (variables)
   int x;
   int y;

public:
   // Declare member function prototypes
   Point(int x = 0, int y = 0);  // Constructor with default values
   int getX() const;
   void setX(int x);
   int getY() const;
   void setY(int y);
   void setXY(int x, int y);
   double getMagnitude() const;
   double getArgument() const;
   void print() const;
};

#endif
```

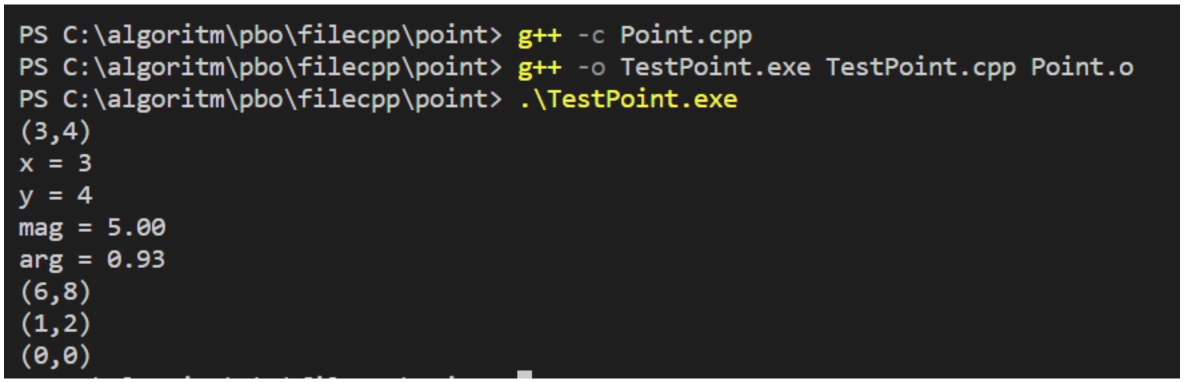
- **Point.cpp**

```
/* The Point class Implementation (Point.cpp) */
#include "Point.h" // user-defined header in the same directory
#include <iostream>
#include <cmath>
using namespace std;

// Constructor (default values can only be specified in the
declaration)
Point::Point(int x, int y) : x(x), y(y) { }  // Use member initializer
list

// Public getter for private data member x
int Point::getX() const {
   return x;
}
// Public setter for private data member x
void Point::setX(int x) {
   this->x = x;
}
// Public getter for private data member y
int Point::getY() const {
   return y;
}
// Public setter for private data member y
void Point::setY(int y) {
   this->y = y;
}
// Public member function to set both x and y
void Point::setXY(int x, int y) {
   this->x = x;
   this->y = y;
}
// Public member function to return the magitude
double Point::getMagnitude() const {
   return sqrt(x*x + y*y);    // sqrt in <cmath>
}
// Public member function to return the argument
double Point::getArgument() const {
   return atan2(y, x);    // atan2 in <cmath>
}
//Public member function to print description about this point
void Point::print() const {
   cout << "(" << x << "," << y << ")" << endl;
}
```

- **TestPoint.cpp**

```
/* A test driver for the Point class (TestPoint.cpp) */
#include <iostream>
#include <iomanip>
#include "Point.h"   // using Point class
using namespace std;

int main() {
   // Construct an instance of Point p1
   Point p1(3, 4);
   p1.print();
   cout << "x = " << p1.getX() << endl;
   cout << "y = " << p1.getY() << endl;
   cout << fixed << setprecision(2);
   cout << "mag = " << p1.getMagnitude() << endl;
   cout << "arg = " << p1.getArgument() << endl;
   p1.setX(6);
   p1.setY(8);
   p1.print();
   p1.setXY(1, 2);
   p1.print();

   // Construct an instance of Point using default constructor
   Point p2;
   p2.print();
}
```

- **Output**



```
PS C:\algoritm\pbo\filecpp\point> g++ -c Point.cpp
PS C:\algoritm\pbo\filecpp\point> g++ -o TestPoint.exe TestPoint.cpp Point.o
PS C:\algoritm\pbo\filecpp\point> .\TestPoint.exe
(3,4)
x = 3
y = 4
mag = 5.00
arg = 0.93
(6,8)
(1,2)
(0,0)
```

4. Time

- **Source Code**

  - **Time.h**

```
/* Header for the Time class (Time.h) */
#ifndef TIME_H
#define TIME_H

class Time {
private:  // private section
   // private data members
   int hour;      // 0 - 23
   int minute;    // 0 - 59
   int second;    // 0 - 59

public:    // public section
   // public member function prototypes
   Time(int h = 0, int m = 0, int s = 0); // Constructor with default
values

   //public getter & setter untuk hour
   int getHour() const;
   void setHour(int h);
   //public getter & setter untuk minute
   int getMinute() const;
   void setMinute(int m);
   //public getter & setter untuk second
   int getSecond() const;
   void setSecond(int s);
   // set hour, minute and second
   void setTime(int h, int m, int s);
   void print() const; // Print a description of this instance in
"hh:mm:ss"
   void nextSecond();  // Increase this instance by one second
};  // need to terminate the class declaration with a semicolon

#endif
```

- **Time.cpp**

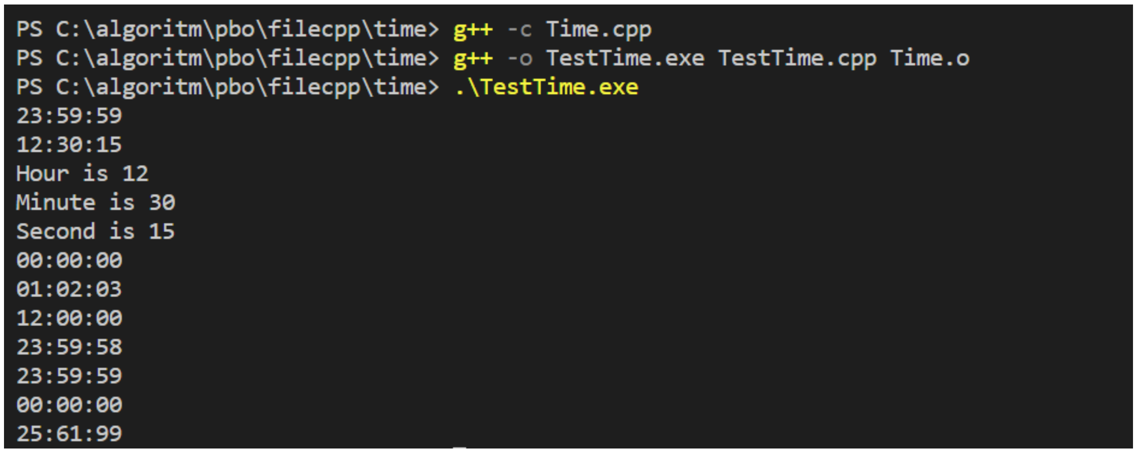
```
/* Implementation for the Time Class (Time.cpp) */
#include <iostream>
#include <iomanip>
#include "Time.h"    // include header of Time class
using namespace std;

// Constructor with default values. No input validation
Time::Time(int h, int m, int s) {
   hour = h;
   minute = m;
   second = s;
}

// public getter untuk hour
int Time::getHour() const {
   return hour;
}
// public setter untuk hour. No input validation
void Time::setHour(int h) {
   hour = h;
}

// public getter untuk minute
int Time::getMinute() const {
   return minute;
}
// public setter untuk minute. No input validation
void Time::setMinute(int m) {
   minute = m;
```

```cpp
}

// public getter untuk second
int Time::getSecond() const {
   return second;
}
// public setter untuk second. No input validation
void Time::setSecond(int s) {
   second = s;
}

// Set hour, minute and second. No input validation
void Time::setTime(int h, int m, int s) {
   hour = h;
   minute = m;
   second = s;
}

// Print this Time instance in the format of "hh:mm:ss", zero filled
void Time::print() const {
   cout << setfill('0');    // zero-filled, need <iomanip>, sticky
   cout << setw(2) << hour  // set width to 2 spaces, need <iomanip>,
non-sticky
        << ":" << setw(2) << minute
        << ":" << setw(2) << second << endl;
}

// Increase this instance by one second
void Time::nextSecond() {
   ++second;
   if (second >= 60) {
      second = 0;
      ++minute;
   }
   if (minute >= 60) {
      minute = 0;
      ++hour;
   }
   if (hour >= 24) {
      hour = 0;
   }
}
```

- **TestTime.cpp**

```
/* Test Driver for the Time class (TestTime.cpp) */
#include <iostream>
#include "Time.h"    // include header of Time class
using namespace std;

int main() {
   Time t1(23, 59, 59);   // Test constructor

   // Test all public member functions
   t1.print();       // 23:59:59
   t1.setHour(12);
   t1.setMinute(30);
   t1.setSecond(15);
   t1.print();       // 12:30:15
   cout << "Hour is "   << t1.getHour()   << endl;
   cout << "Minute is " << t1.getMinute() << endl;
   cout << "Second is " << t1.getSecond() << endl;

   // Test constructor with default values for hour, minute and second
   Time t2;
   t2.print();  // 00:00:00
   t2.setTime(1, 2, 3);
   t2.print();  // 01:02:03

   Time t3(12); // Use default values for minute and second
   t3.print();  // 12:00:00

   // Test nextSecond()
   Time t4(23, 59, 58);
   t4.print();
   t4.nextSecond();
   t4.print();
   t4.nextSecond();
   t4.print();

   // No input validation
   Time t5(25, 61, 99); // values out of range
   t5.print();  // 25:61:99
}
```

- **Output**



```
PS C:\algoritm\pbo\filecpp\time> g++ -c Time.cpp
PS C:\algoritm\pbo\filecpp\time> g++ -o TestTime.exe TestTime.cpp Time.o
PS C:\algoritm\pbo\filecpp\time> .\TestTime.exe
23:59:59
12:30:15
Hour is 12
Minute is 30
Second is 15
00:00:00
01:02:03
12:00:00
23:59:58
23:59:59
00:00:00
25:61:99
```

5. Account

- **Source Code**

  - **Account.h**

```
/* Header for Account class (Account.h) */
#ifndef ACCOUNT_H
#define ACCOUNT_H

class Account {
private:
   int accountNumber;
   double balance;

public:
   Account(int accountNumber, double balance = 0.0);
   int getAccountNumber() const;
   double getBalance() const;
   void setBalance(double balance);
   void credit(double amount);
   void debit(double amount);
   void print() const;
};

#endif
```

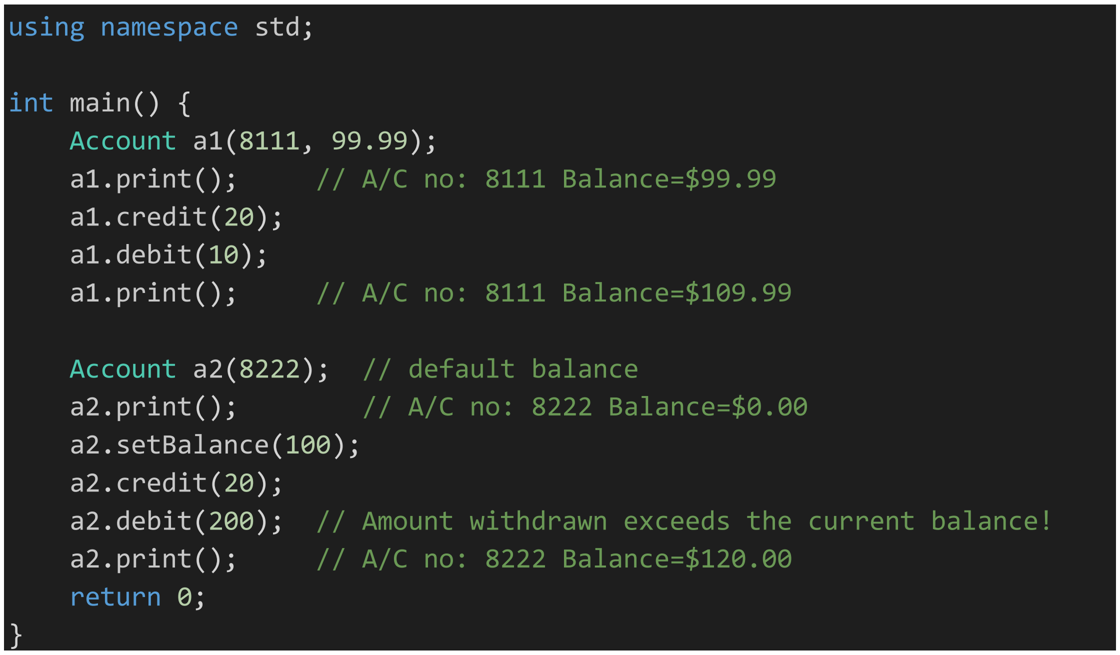
  - **Account.cpp**

```
/* Implementation for the Account class (Account.cpp) */
#include <iostream>
#include <iomanip>
#include "Account.h"
```

```
using namespace std;

// Constructor
Account::Account(int no, double b) : accountNumber(no), balance(b) { }

// Public getter untuk accountNumber
int Account::getAccountNumber() const {
   return accountNumber;
}

// Public getter untuk balance
double Account::getBalance() const {
   return balance;
}
// Public setter untuk balance
void Account::setBalance(double b) {
   balance = b;
}

// Adds the given amount to the balance
void Account::credit(double amount) {
   balance += amount;
}

// Subtract the given amount from the balance
void Account::debit(double amount) {
   if (amount <= balance) {
      balance -= amount;
   } else {
      cout << "Amount withdrawn exceeds the current balance!" << endl;
   }
}

// Print description for this Account instance
void Account::print() const {
   cout << fixed << setprecision(2);
   cout << "A/C no: " << accountNumber << " Balance=$" << balance <<
endl;
}
```

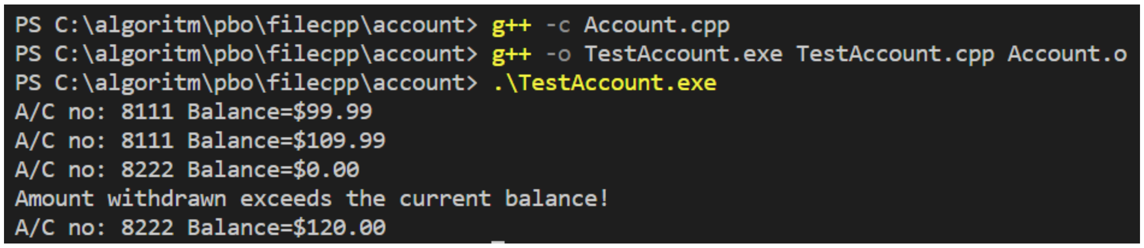
- **TestAccount.cpp**

```
/* Test Driver for Account class (TestAccount.cpp) */
#include <iostream>
#include "Account.h"
```



```cpp
using namespace std;

int main() {
    Account a1(8111, 99.99);
    a1.print();     // A/C no: 8111 Balance=$99.99
    a1.credit(20);
    a1.debit(10);
    a1.print();     // A/C no: 8111 Balance=$109.99

    Account a2(8222);  // default balance
    a2.print();        // A/C no: 8222 Balance=$0.00
    a2.setBalance(100);
    a2.credit(20);
    a2.debit(200);  // Amount withdrawn exceeds the current balance!
    a2.print();     // A/C no: 8222 Balance=$120.00
    return 0;
}
```

- **Output**



```
PS C:\algoritm\pbo\filecpp\account> g++ -c Account.cpp
PS C:\algoritm\pbo\filecpp\account> g++ -o TestAccount.exe TestAccount.cpp Account.o
PS C:\algoritm\pbo\filecpp\account> .\TestAccount.exe
A/C no: 8111 Balance=$99.99
A/C no: 8111 Balance=$109.99
A/C no: 8222 Balance=$0.00
Amount withdrawn exceeds the current balance!
A/C no: 8222 Balance=$120.00
```