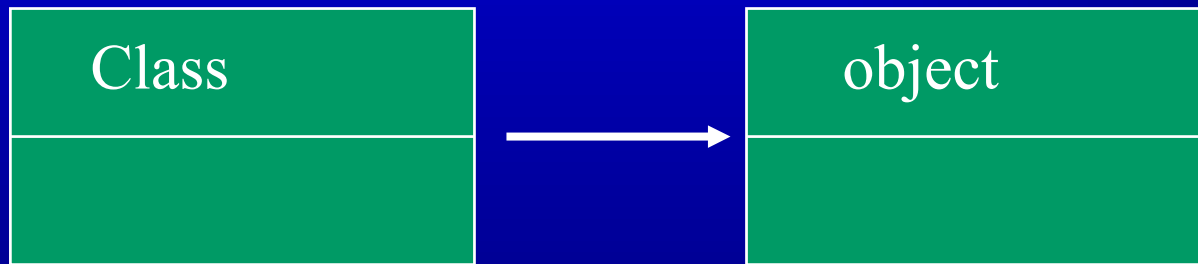# Chapter 5
Programming with Objects and Classes

☞ OO Programming Concepts

☞ Declaring and Creating Objects

☞ Constructors

☞ Modifiers (`public`, `private` and `static`)

☞ Instance and Class Variables and Methods

☞ Scope of Variables

☞ Use the this Keyword

☞ Analyze Relationships among Classes

☞ Case Studies (`Mortgage` class and `Rational` class)

☞ The Java API and Core Java classes

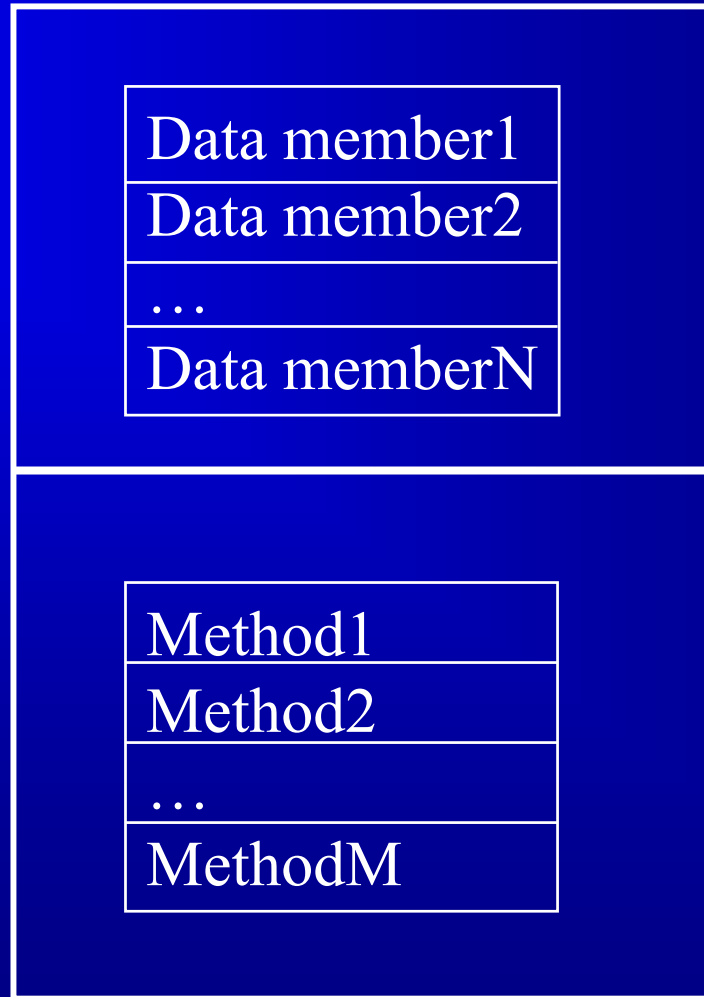☞ Processing Strings (`String`, `StringBuffer`, and `StringTokenizer`)

OO Programming Concepts

- Classes are blueprints or structures for defining objects
- Each object is an instance of class
- In Java, each object has data which are used to describe properties, and methods which are used to describe behaviors
- Each Java file is a class

Class

→

object

OO Programming Concepts

Class construction

Data member1

Data member2

…

Data memberN

properties

Method1

Method2

…

MethodM

behaviors

Class and Objects

Circle class

Circle
radius
findArea
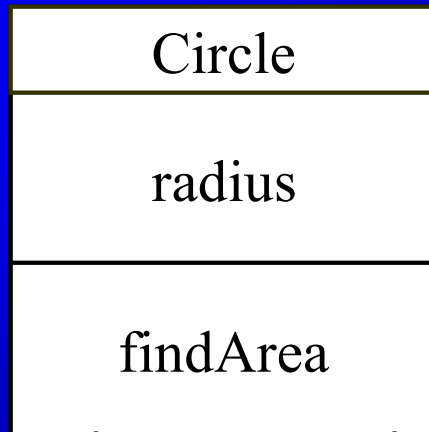
Graphical notation for classes

new Circle()

new Circle()

circle1: Circle
radius = 2

...

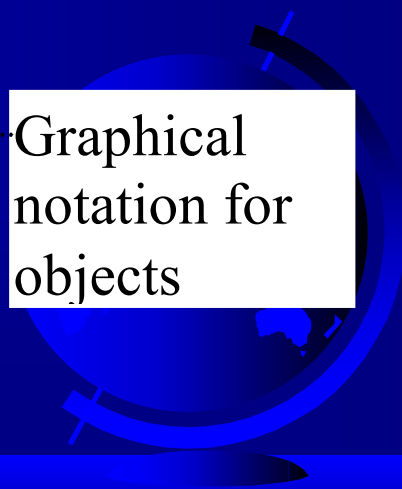| circlen: Circle |
|---|
| radius = 5 |
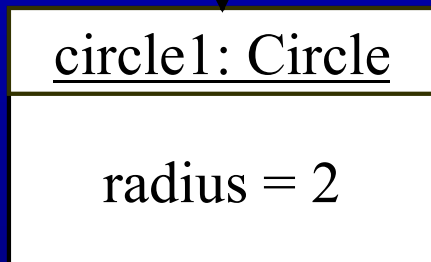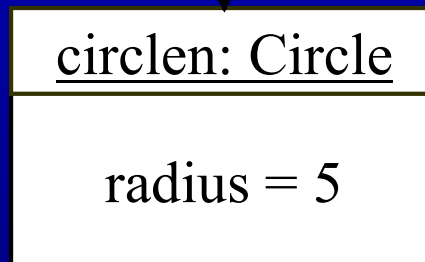
Graphical notation for objects

Class Declaration

```
class Circle
{
  double radius = 1.0;

  double findArea()
  {
    return radius*radius*3.14159;
  }
}
```

Declaring Objects

```
ClassName objectName;
```

Example:

```
Circle myCircle;
```

Creating Objects

```
objectName = new ClassName();
```

Example:

```
myCircle = new Circle();
```

Declaring/Creating Objects in a Single Step

```
ClassName objectName = new
   ClassName();
```

Example:

```
Circle myCircle = new Circle();
```

Differences Between Variables of Primitive Data Types and Object Types

| Primitive type | int i = 1 | i | 1 |

| Object type | Circle c | c | reference |

Created using new Circle(5)

c: Circle

radius = 5

Copying Variables of Primitive Data Types and Object Types

Primitive type assignment
i = j

Object type assignment
c1 = c2

Before:

After:

i	1

i	2

j	2

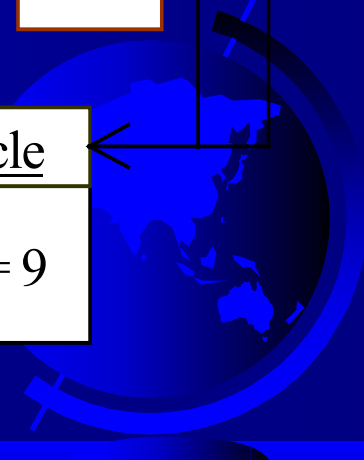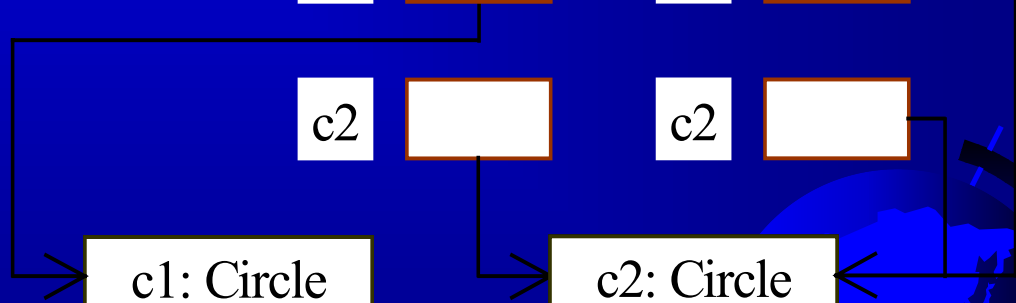| j	2

Before:

After:

c1

c1

c2

c2

c1: Circle

radius = 5

c2: Circle

radius = 9

Garbage Collection

☞ After c1=c2, c1 object is known as garbage

☞ Java runtime system detects garbage and automatically reclaims the space it occupies

☞ Garbage collection

Accessing Objects

☞ Referencing the object's data:

`objectName.data`

`myCircle.radius` ← Instance variable

☞ Referencing the object's method:

`objectName.method`

← Instance method

`myCircle.findArea()`

Example 5.1
Using Objects

☞ Objective: Demonstrate creating objects, accessing data, and using methods.

```java
// TestCircle.java: Demonstrate creating and using an object
public class TestCircle
{
  // Main method
  public static void main(String[] args)
  {
    Circle myCircle = new Circle();  // Create a Circle object
    System.out.println("The area of the circle of radius "
      + myCircle.radius + " is " + myCircle.findArea());

    Circle lynnCircle =new Circle();
```

```
System.out.println(lynnCircle.radius);
  System.out.println( lynnCircle.findArea() );

  Circle c1, c2, c3,c4,c5;
  c1=new Circle();   c2=new Circle();   c3=new Circle();
    c4=new Circle();   c5=new Circle();


  System.out.println(
  c1.radius+c2.radius
  +c3.radius+c4.radius
    +c5.radius);
 }
}
```

```
class Circle
{
  double radius=1.0;

  // Find area of this circle
  public double findArea()
  {
    return radius*radius*3.14159;
  }

}
```

Constructors

```
Circle(double r)
{
  radius = r;
}


Circle()   //default constructor
{
  radius = 1.0;
}


myCircle = new Circle(5.0);
```

Example 5.2
Using Constructors

☞ Objective: Discuss the role of constructors and use them to create objects.

```java
// TestCircleWithConstructors.java: Demonstrate constructors
public class TestCircleWithConstructors
{
  // Main method
  public static void main(String[] args)
  {
    // Create a Circle with radius 5.0
    Circle myCircle = new Circle(5.0);
    System.out.println("The area of the circle of radius "
      + myCircle.radius + " is " + myCircle.findArea());

    // Create a Circle with default radius
    Circle yourCircle = new Circle();
    System.out.println("The area of the circle of radius "
      + yourCircle.radius + " is " + yourCircle.findArea());
  }
}
```

```
// Circle with two constructors
class Circle
{
  double radius;

  // Default constructor
  Circle()
  {
    radius = 1.0;
  }

  // Construct a circle with a specified radius
  Circle(double r)
  {
    radius = r;
  }
}
```
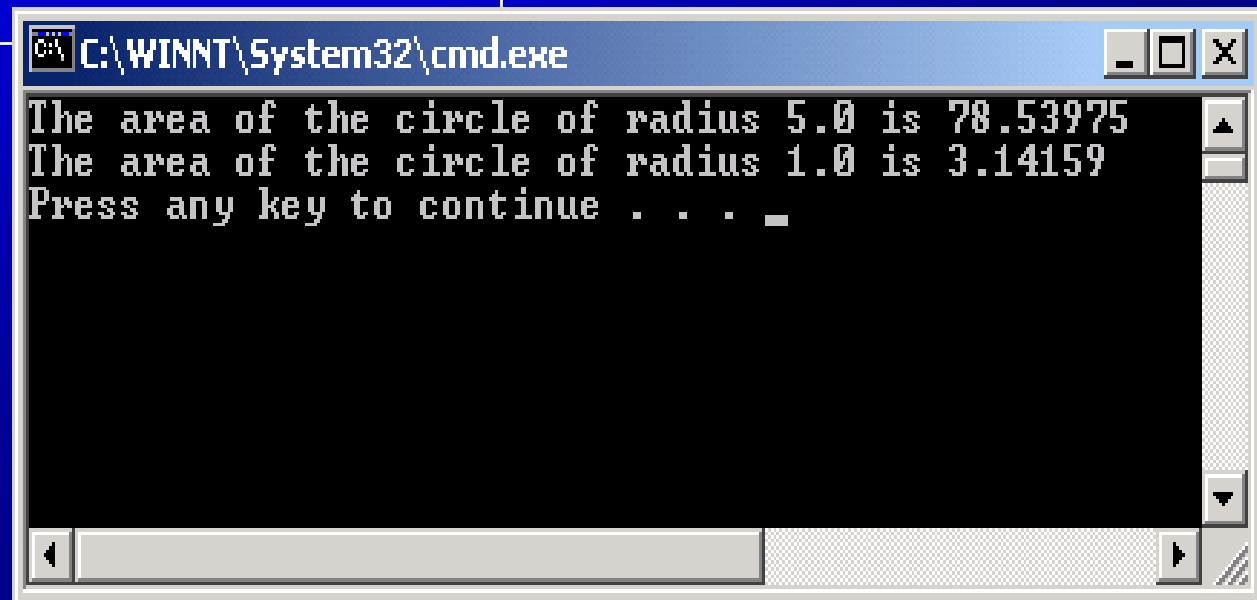
```
// Find area of this circle
double findArea()
{
  return radius*radius*3.14159;
}
}
```



```
C:\WINNT\System32\cmd.exe

The area of the circle of radius 5.0 is 78.53975
The area of the circle of radius 1.0 is 3.14159
Press any key to continue . . . _
```

Passing Objects to Methods

☞ Passing by value

☞ Passing by reference (passing objects)

Example 5.3 Passing Objects as Arguments

```java
// TestPassingObject.java: Demonstrate passing objects to methods
public class TestPassingObject
{
  // Main method
  public static void main(String[] args)
  {
    // Create a Circle object with default radius 1
    Circle myCircle = new Circle();

    // Print areas for radius 1, 2, 3, 4, and 5.
    int n = 5;                          ←——  Passing by value
    printAreas(myCircle, n);
                          ←——  Passing by reference
    // See myCircle.radius and times
    System.out.println("\n" + "Radius is " + myCircle.radius);
    System.out.println("n is " + n);
  }
```

```java
// Print a table of areas for radius
public static void printAreas(Circle c, int times)
{
  System.out.println("Radius \t\tArea");
  while (times >= 1)
  {
    System.out.println(c.radius + "\t\t" + c.findArea());
    c.radius++;
    times--;
  }
}
```

```
C:\WINNT\System32\cmd.exe                           _ □ ×

Radius              Area
1.0                 3.14159
2.0                 12.56636
3.0                 28.27431
4.0                 50.26544
5.0                 78.53975

Radius is 6.0
n is 5
Press any key to continue . . . _
```

Visibility Modifiers and Accessor Methods

By default, the class, variable, or data can be accessed by any class in the same package.

☞ public

The class, data, or method is visible to any class in any package.

☞ private

The data or methods can be accessed only by the declaring class.

The getter and setter accessor methods are used to read and modify private properties.

Example 5.4
Using the `private` Modifier and Accessor Methods

In this example, private data are used for the radius and the accessor methods getRadius and setRadius are provided for the clients to retrieve and modify the radius.

```java
// TestCircleWithPrivateModifier.java: Demonstrate private modifier
public class TestCircleWithPrivateModifier
{
  // Main method
  public static void main(String[] args)
  {
    // Create a Circle with radius 5.0
    CircleSecond myCircle = new CircleSecond(5.0);
    System.out.println("The area of the circle of radius "
      + myCircle.getRadius() + " is " + myCircle.findArea());

    // Increase myCircle's radius by 10%
    myCircle.setRadius(myCircle.getRadius()*1.1);
    System.out.println("The area of the circle of radius "
      + myCircle.getRadius() + " is " + myCircle.findArea());
  }
}
```

```java
// Declare class Circle with private radius and accessor methods
public class CircleSecond
{
  private double radius;

  // Default constructor
  public CircleSecond()
  {
    radius = 1.0;
  }


  // Construct a circle with a specified radius
  public CircleSecond(double r)
  {
    radius = r;
  }
```

```java
// Getter method for radius
public double getRadius()
{
  return radius;
}


// Setter method for radius
public void setRadius(double newRadius)
{
  radius = newRadius;
}


// Find the circle area
public double findArea()
{
  return radius*radius*3.14159;
}
}
```
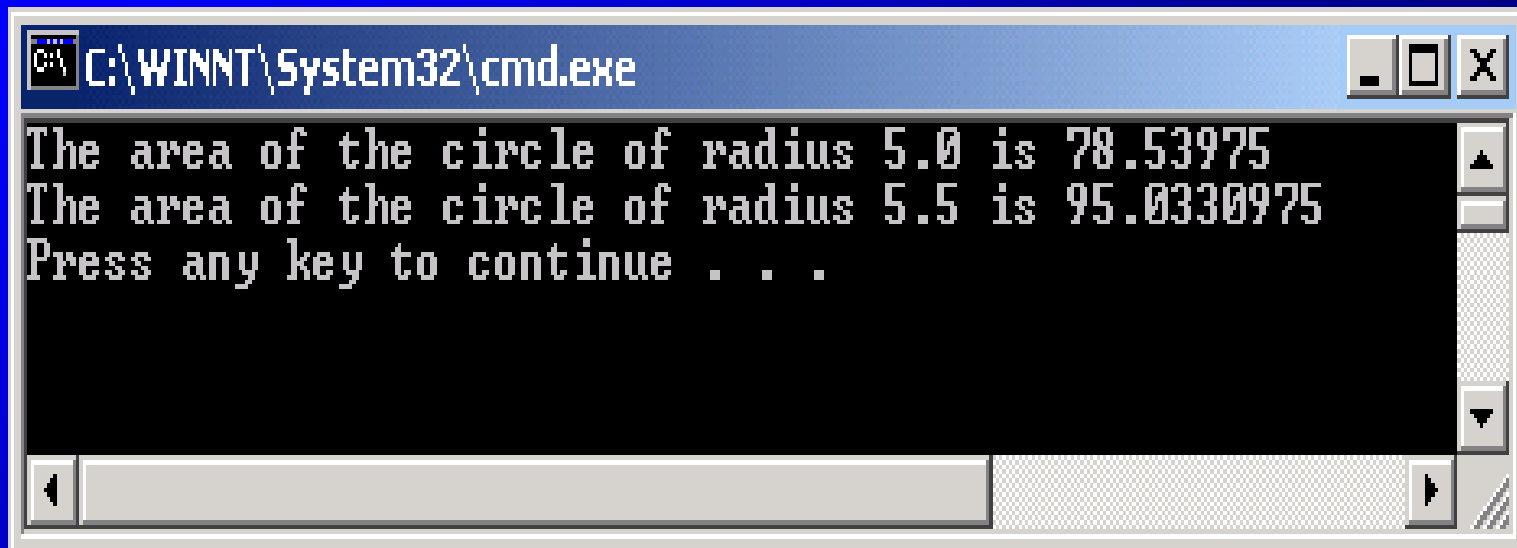
```
C:\WINNT\System32\cmd.exe

The area of the circle of radius 5.0 is 78.53975
The area of the circle of radius 5.5 is 95.0330975
Press any key to continue . . .
```

Instance Variables and Methods

Instance variables belong to a specific instance.

Instance methods are invoked by an instance of the class.

myCircle.radius

myCircle.findArea()

Class Variables, Constants, and Methods

- Class variables are shared by all the instances of the class.

- Class methods are not tied to a specific object.

- Class constants are final variables shared by all the instances of the class.

- Class variables, constants, and methods are used with class name, such as Math.pow(2,3), Math.PI
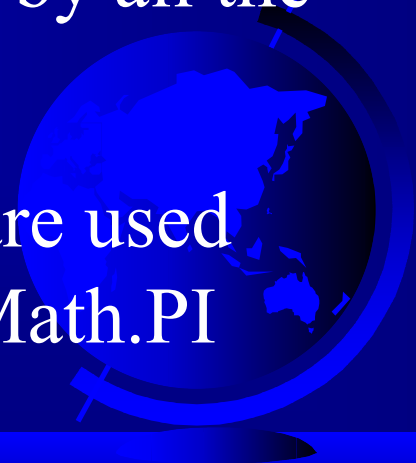
Class Variables, Constants, and Methods (Cont.)

To declare class variables, constants, and methods, use the static modifier.

```
static int numOfObject;

public final static double PI=3.14159265358979323846;

static returnValueType staticMethod(…)
```

Class Variables, Constants, and Methods (Cont.)

Notation:
+: public variables or methods
-: private variables or methods
underline: static variables or metods

Memory

radius is an instance variable, and numOfObjects is a class variable

Circle
-radius -<u>numOfObjects</u>
+getRadius +setRadius +<u>getNumOfObjects</u> +findArea

instantiate

circle1:Circle
-radius = 1 -<u>numOfObjects</u> = 2

instantiate

circle2:Circle
-radius = 5 -<u>numOfObjects</u> = 2

1   radius

2   numOfObjects

5   radius
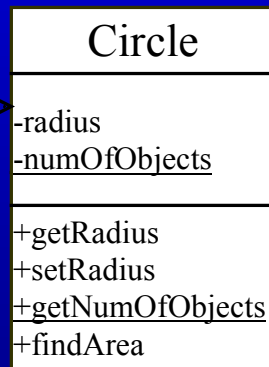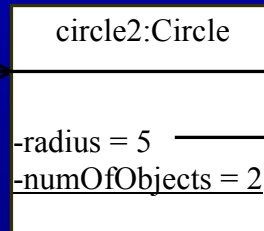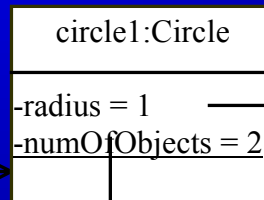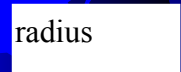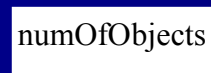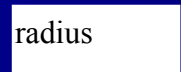
Example 5.5
Using Instance and Class Variables and Method

Objective: Demonstrate the roles of instance and class variables and their uses. This example adds a class variable numOfObjects to track the number of Circle objects created.

```java
// TestInstanceAndClassVariable.java: Demonstrate using instance
// and class variables
public class TestInstanceAndClassVariable
{
  // Main method
  public static void main(String[] args)
  {
    // Create circle1
    Circle circle1 = new Circle();

    // Display circle1 BEFORE circle2 is created
    System.out.println("Before creating circle2");
    System.out.print("circle1 is : ");
    printCircle(circle1);

    // Create circle2
    Circle circle2 = new Circle(5);
```

```java
// Change the radius in circle1
    circle1.setRadius(9);
    // Display circle1 and circle2 AFTER circle2 was created
    System.out.println("\nAfter creating circle2 and modifying " +
        "circle1's radius to 9");
    System.out.print("circle1 is : ");
    printCircle(circle1);
    System.out.print("circle2 is : ");
    printCircle(circle2);
}
// Print circle information
public static void printCircle(Circle c)
{
    System.out.println("radius (" + c.getRadius() +
        ") and number of Circle objects (" +
        c.getNumOfObjects() + ")");
}
}
```

```java
// Circle.java: Circle class with instance and class variables
class Circle                    //innner class
{
  private double radius;
  private static int numOfObjects = 0;  // Class variable
  // Default constructor
  public Circle()
  {
    radius = 1.0;
    numOfObjects++;
  }
  // Construct a circle with a specified radius
  public Circle(double r)
  {
    radius = r;
    numOfObjects++;
  }
}
```
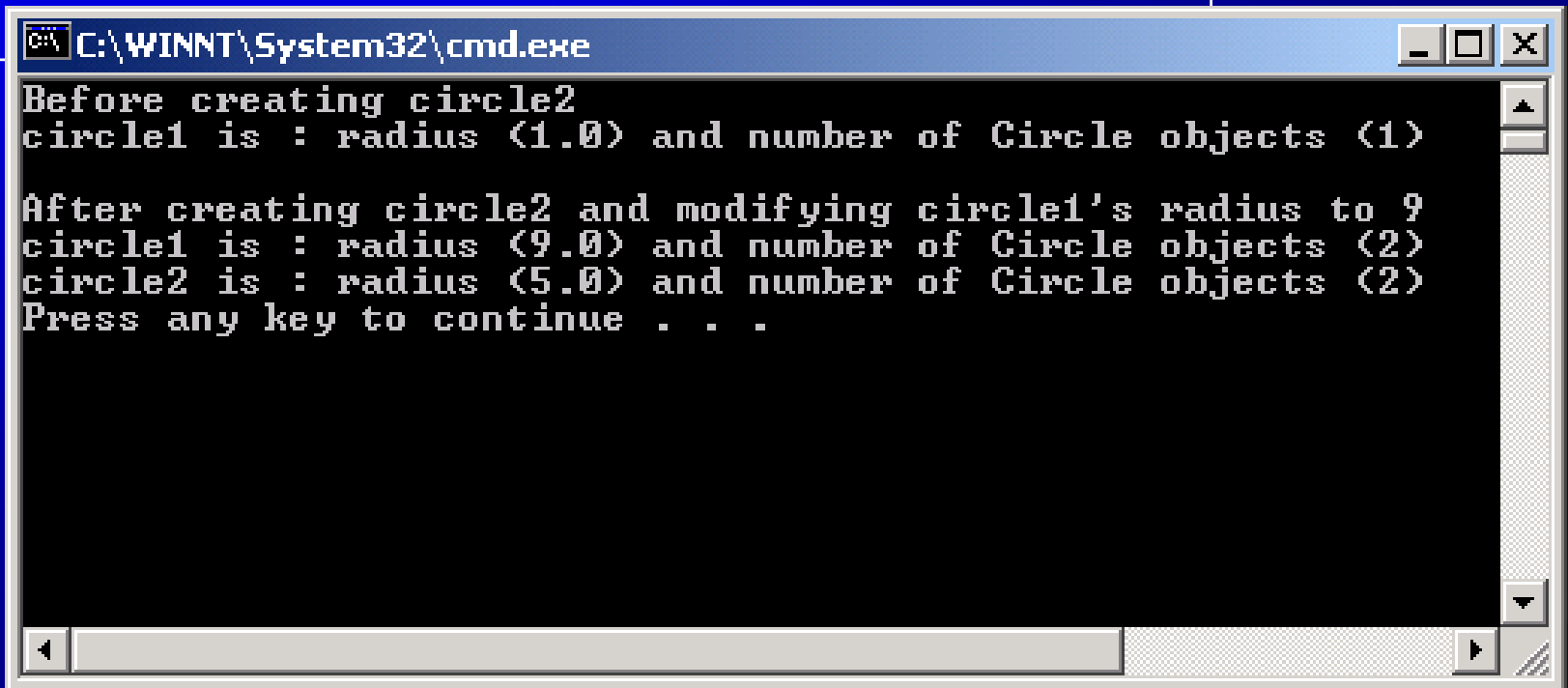
```java
// Getter method for radius
  public double getRadius()
  {
    return radius;
  }


  // Setter method for radius
  public void setRadius(double newRadius)
  {
    radius = newRadius;
  }


  // Getter method for numOfObjects
  public static int getNumOfObjects()
  {
    return numOfObjects;
  }
```

Class method

```java
// Find circle area
  public double findArea()
  {
    return radius*radius*Math.PI;

  }
}
```

```
C:\WINNT\System32\cmd.exe

Before creating circle2
circle1 is : radius (1.0) and number of Circle objects (1)

After creating circle2 and modifying circle1's radius to 9
circle1 is : radius (9.0) and number of Circle objects (2)
circle2 is : radius (5.0) and number of Circle objects (2)
Press any key to continue . . .
```

Scope of Variables

☞ The scope of instance and class variables is the entire class. They can be declared anywhere inside a class. They are global variables

☞ The scope of a local variable starts from its declaration and continues to the end of the block that contains the variable. A local variable must be declared before it can be used.

```
class Circle
{
  private double radius;                    ←———  Global variable

  public double findArea()
  {
   double result;                    ←———  Local variable
    result= radius*radius*Math.PI;
   return result;
  }
}
```

```java
class Foo
{
  int x=0;
  int y=0;

  Foo()
  {
  }

  void p()
  {
  int x=1;
  System.out.println(x);
  System.out.println(y);
  }
}
```

Local variable

```java
for (int j=0; j<20;j++)
{
…

}
```

Relationships Among Classes

☞Association (shared)

☞Aggregation (has or composes)

☞Inheritance (developed from)

Association

Association represents a general binary relationship that describes an activity between two classes.

Student	5..60	Take	▶	*	Course	0..3	Teach	◀	1	Faculty
									Teacher	
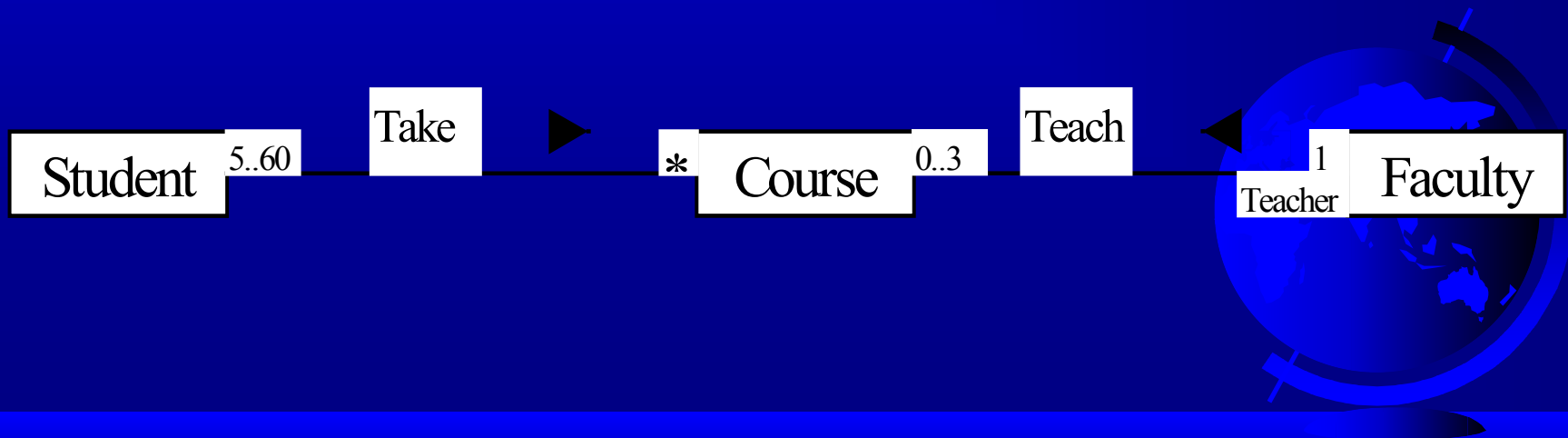
Aggregation

Aggregation is a special form of association, which represents an ownership relationship between two classes. Aggregation models the relationship like has-a, part-of, owns, and employed-by.

Inheritance

Inheritance models the is-a relationship between two classes.
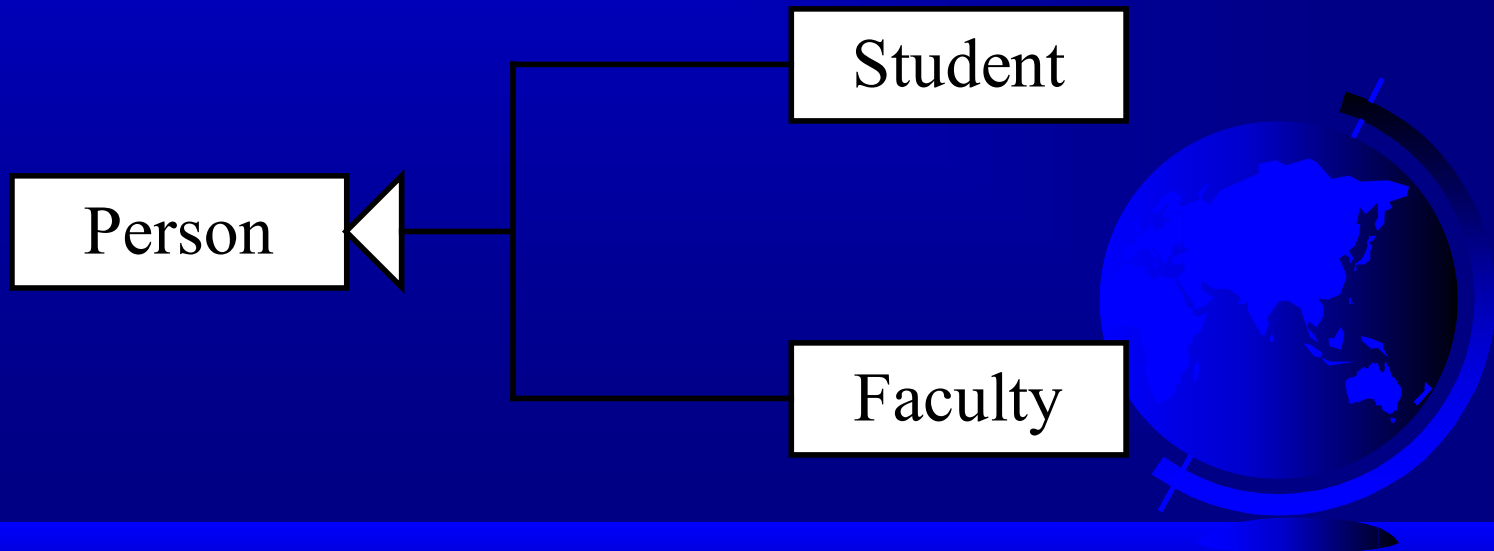
Class Abstraction

  Class abstraction means to separate class implementation from the use of the class. The creator of the class provides a description of the class and let the user know how the class can be used. The user of the class does not need to know how the class is implemented. The detail of implementation is encapsulated and hidden from the user.

Class Design

1. Identify classes for the system.

2. Describe attributes and methods in each class.

3. Establish relationships among classes.

4. Create classes.

Example 5.6 Borrowing Mortgages

Name

Name	1
firstname -mi -lastname	
+getFirstname +getMi +getLastname +setFirstname +setMi +setLastname +getFullname	

Has

Borrower

1	Borrower	1
	-name -address -mortgage	
	+getName +getAddress +getMortgage +setName +setAddress +setMortgage	

Live

Address

1	Address
	-street -city -state -zip
	+getStreet +getCity +getState +getZip +setStreet +setCity +setState +setZip +getAddress

Borrow

Mortgage

Mortgage
-annualInterestRate -numOfYears -loanAmount
+getAnnualInterestRate +getNumOfYears +getLoanAmount +setAnnualInterestRate +setNumOfYears +setLoanAmount +monthlyPayment +totalPayment

Example 5.6 Borrowing Mortgages, (Cont.)

The following is a test program that uses the classes `Name`, `Address`, `Borrower`, and `Mortgage`.

```java
// Name.java: Encapsulate name information
public class Name
{
  private String firstname;
  private String mi;
  private String lastname;

  // Default constructor
  public Name()
  {
  }
  // Construct a name with firstname, mi, and lastname
  public Name(String firstname, String mi, String lastname)
  {
    this.firstname = firstname;
    this.mi = mi;
    this.lastname = lastname;
  }
```

```java
// Getter method for firstname
public String getFirstName()
{
  return firstname;
}


// Setter method for firstname
public void setFirstname(String firstname)
{
  this.firstname = firstname;
}


// Getter method for middlename initial
public String getMi()
{
  return mi;
}
```

```java
// Setter method for middlename initial
public void setMi(String mi)
{
  this.mi = mi;
}


// Getter method for lastname
public String getLastname()
{
  return lastname;
}


// Setter method for lastname
public void setLastname(String lastname)
{
  this.lastname = lastname;
}
```

```
// Obtain full name
public String getFullname()
{
  return firstname + ' ' + mi + ' ' + lastname;
}
}
```

```java
// Address.java: Encapsulate address information
public class Address
{
  private String street;
  private String city;
  private String state;
  private String zip;

  // Default constructor
  public Address()
  {
  }
```

```java
// Create address with street, city, state, and zip
public Address(String street, String city,
    String state, String zip)
{
    this.street = street;
    this.city = city;
    this.state = state;
    this.zip = zip;
}

// Getter method for street
public String getStreet()
{
    return street;
}
```

```java
// Setter method for street
public void setStreet(String street)
{
  this.street = street;
}

// Getter method for city
public String getCity()
{
  return city;
}

// Setter method for city
public void setCity(String city)
{
  this.city = city;
}
```

```java
// Getter method for state
public String getState()
{
  return state;
}

// Setter method for state
public void setState(String state)
{
  this.state = state;
}
// Getter method for zip
public String getZip()
{
  return zip;
}
```

```java
// Setter method for zip
public void setZip(String zip)
{
  this.zip = zip;
}

// Get full address
public String getFullAddress()
{
  return street + '\n' + city + ", " + state + ' ' + zip + '\n';
}
}
```

```java
// Mortgage.java: Encapsulate mortgage information
public class Mortgage
{
  private double annualInterestRate;
  private int numOfYears;
  private double loanAmount;

  // Default constructor
  public Mortgage()
  {
  }
```

```java
// Construct a mortgage with specified annual interest rate,
// number of years and loan amount
public Mortgage(double annualInterestRate, int numOfYears,
  double loanAmount)
{
  this.annualInterestRate = annualInterestRate;
  this.numOfYears = numOfYears;
  this.loanAmount = loanAmount;
}

// Getter method for annualInterestRate
public double getAnnualInterestRate()
{
  return annualInterestRate;
}
```

```java
// Setter method for annualInterestRate
public void setAnnualInterestRate(double annualInterestRate)
{
  this.annualInterestRate = annualInterestRate;
}

// Getter method for numOfYears
public int getNumOfYears()
{
  return numOfYears;
}

// Setter method for numOfYears
public void setNumOfYears(int numOfYears)
{
  this.numOfYears = numOfYears;
}
```

```java
// Getter method for loanAmount
public double getLoanAmount()
{
  return loanAmount;
}
// Setter method for loanAmount
public void setLoanAmount(double loanAmount)
{
  this.loanAmount = loanAmount;
}
// Find monthly payment
public double monthlyPayment()
{
  double monthlyInterestRate = annualInterestRate/1200;
  return loanAmount*monthlyInterestRate/
    (1 - (Math.pow(1/(1 + monthlyInterestRate), numOfYears*12)));
}
```

```java
// Find total payment
  public double totalPayment()
  {
    return monthlyPayment()*numOfYears*12;
  }
}
```

```java
// Borrower.java: Encapsulate borrower information
public class Borrower
{
  private Name name;
  private Address address;
  private Mortgage mortgage;

  // Default constructor
  public Borrower()
  {
  }


  // Getter method for name
  public Name getName()
  {
    return name;
  }
}
```
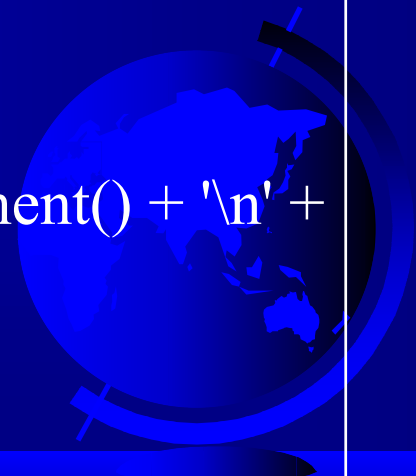
```java
// Setter method for name
public void setName(Name name)
{
  this.name = name;
}


// Getter method for address
public Address getAddress()
{
  return address;
}
// Setter method for address
public void setAddress(Address address)
{
  this.address = address;
}
```

```java
// Getter method for mortgage
public Mortgage getMortgage()
{    return mortgage; }
// Setter method for mortgage
public void setMortgage(Mortgage mortgage)
{
  this.mortgage = mortgage;
}
// String representation for borrower
public String toString()
{
  return '\n' + name.getFullname() + '\n' +
    address.getFullAddress() + '\n' +
    "Monthly payment is " + mortgage.monthlyPayment() + '\n' +
    "Total payment is " + mortgage.totalPayment();
}
}
```

```java
// BorrowMortgage.java: Demonstrate using the classes Borrower
// Name, Address, and Mortgage
public class BorrowMortgage
{
  // Main method
  public static void main(String[] args)
  {
    // Create one borrower
    Borrower borrower = new Borrower();

    // Enter the information for the borrower

    // 1. Enter name
    Name name = new Name();
    System.out.print("Enter first name: ");
    String firstname = MyInput.readString();
    name.setFirstname(firstname);
```

```java
System.out.print("Enter mi: ");
String mi = MyInput.readString();
name.setMi(mi);

System.out.print("Enter last name: ");
String lastname = MyInput.readString();
name.setLastname(lastname);

// 2. Enter address
Address address = new Address();
System.out.print("Enter street: ");
String street = MyInput.readString();
address.setStreet(street);
```

```java
System.out.print("Enter city: ");
  String city = MyInput.readString();
  address.setCity(city);

  System.out.print("Enter state: ");
  String state = MyInput.readString();
  address.setState(state);

  System.out.print("Enter zip: ");
  String zip = MyInput.readString();
  address.setZip(zip);

  // 3. Enter mortgage information
  Mortgage mortgage = new Mortgage();
  System.out.print("Enter annual interest rate (i.e. 7.25): ");
  double annualInterestRate = MyInput.readDouble();
  mortgage.setAnnualInterestRate(annualInterestRate);
```

```java
System.out.print("Enter number of years: ");
int numOfYears = MyInput.readInt();
mortgage.setNumOfYears(numOfYears);

System.out.print("Enter loan amount: ");
double loanAmount = MyInput.readDouble();
mortgage.setLoanAmount(loanAmount);

// 4. Set values to the borrower
borrower.setName(name);
borrower.setAddress(address);
borrower.setMortgage(mortgage);

// Print mortgage information
System.out.println(borrower.toString());
    }
}
```

```
C:\WINNT\System32\cmd.exe

Enter first name: Jun
Enter mi: Enter last name: Ni
Enter street: 322 Denvenport
Enter city: Iowa City
Enter state: IA
Enter zip: 52240
Enter annual interest rate (i.e. 7.25): 4.6
Enter number of years: 10
Enter loan amount: 12321

Jun    Ni
322 Denvenport
Iowa City, IA 52240

Monthly payment is 128.2876426806873
Total payment is 15394.517121682476
Press any key to continue . . . _
```
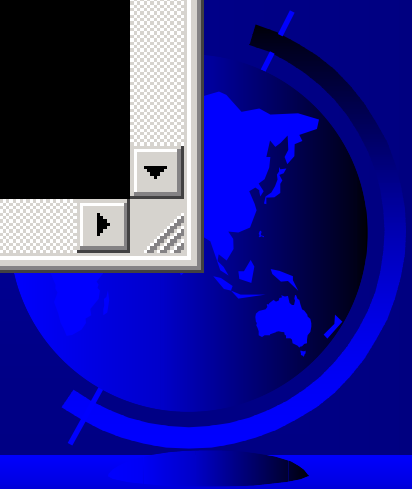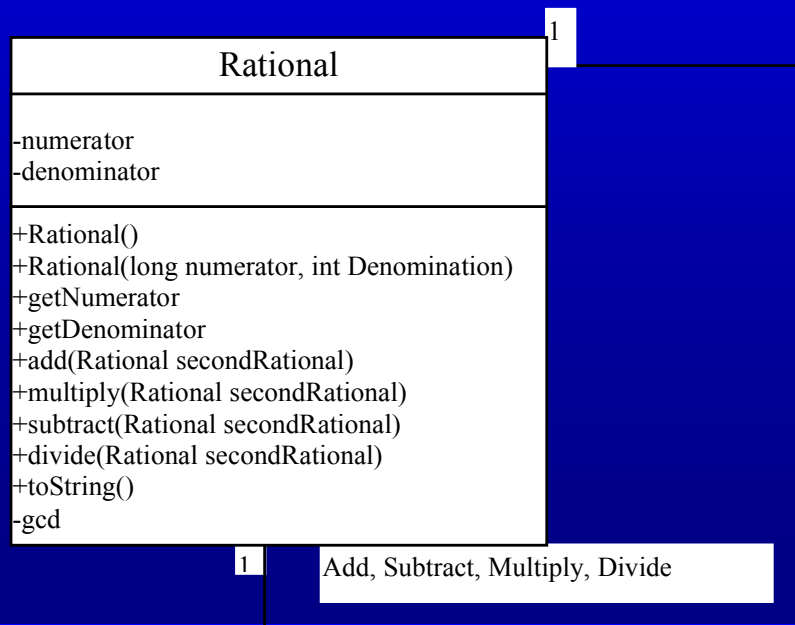
Example 5.7
Using the `Rational` Class

Objective: Define a class for rational numbers that provides constructors and addition, subtraction, multiplication, and division methods.

Rational
-numerator -denominator
+Rational() +Rational(long numerator, int Denomination) +getNumerator +getDenominator +add(Rational secondRational) +multiply(Rational secondRational) +subtract(Rational secondRational) +divide(Rational secondRational) +toString() -gcd

1

1   Add, Subtract, Multiply, Divide

```java
/ Rational.java: Define a rational number and its associated
// operations such as add, subtract, multiply, and divide
public class Rational
{
  // Data fields for numerator and denominator
  private long numerator = 0;
  private long denominator = 1;

  // Default constructor
  public Rational()
  {
    this(0, 1);
  }
```

```java
// Construct a rational with specified numerator and denominator
public Rational(long numerator, long denominator)
{
  long gcd = gcd(numerator, denominator);
  this.numerator = numerator/gcd;
  this.denominator = denominator/gcd;
}

// Find GCD of two numbers
private long gcd(long n, long d)
{
  long t1 = Math.abs(n);
  long t2 = Math.abs(d);
  long remainder = t1%t2;
```

```
while (remainder != 0)
  {
    t1 = t2;
    t2 = remainder;
    remainder = t1%t2;
  }

  return t2;
}

// Getter method for numerator
public long getNumerator()
{
  return numerator;
}
```

```java
public long getDenominator()
{
  return denominator;
}

// Add a rational number to this rational
public Rational add(Rational secondRational)
{
  long n = numerator*secondRational.getDenominator() +
    denominator*secondRational.getNumerator();
  long d = denominator*secondRational.getDenominator();
  return new Rational(n, d);
}
```

```java
// Subtract a rational number from this rational
public Rational subtract(Rational secondRational)
{
  long n = numerator*secondRational.getDenominator()
    - denominator*secondRational.getNumerator();
  long d = denominator*secondRational.getDenominator();
  return new Rational(n, d);
}

// Multiply a rational number to this rational
public Rational multiply(Rational secondRational)
{
  long n = numerator*secondRational.getNumerator();
  long d = denominator*secondRational.getDenominator();
  return new Rational(n, d);
}
```

```java
// Divide a rational number from this rational
public Rational divide(Rational secondRational)
  throws RuntimeException
{
  if (secondRational.getNumerator() == 0)
    throw new RuntimeException("Denominator cannot be zero");

  long n = numerator*secondRational.getDenominator();
  long d = denominator*secondRational.getNumerator();
  return new Rational(n, d);
}
```

```java
// Override the toString() method
public String toString()
{
  if (denominator == 1)
    return numerator + "";
  else
    return numerator + "/" + denominator;
}
}
```

```java
// TestRationalClass.java: Demonstrate using the Rational class
public class TestRationalClass
{
  // Main method
  public static void main(String[] args)
  {
    // Create and initialize two rational numbers r1 and r2.

        int n1,d1, n2,d2;
        System.out.print("Please enter an integer:");
        n1=MyInput.readInt();
        System.out.print("Please enter an integer:");
        d1=MyInput.readInt();
        System.out.print("Please enter an integer:");
        n2=MyInput.readInt();
        System.out.print("Please enter an integer:");
        d2=MyInput.readInt();
```

```java
Rational r1 = new Rational(n1, d1);
Rational r2 = new Rational(n2, d2);

// Display results
System.out.println(r1.toString() + " + " + r2.toString() +
  " = " + (r1.add(r2)).toString());
System.out.println(r1.toString() + " - " + r2.toString() +
  " = " + (r1.subtract(r2)).toString());
System.out.println(r1.toString() + " * " + r2.toString() +
  " = " + (r1.multiply(r2)).toString());
System.out.println(r1.toString() + " / " + r2.toString() +
  " = " + (r1.divide(r2)).toString());
  }
}
```
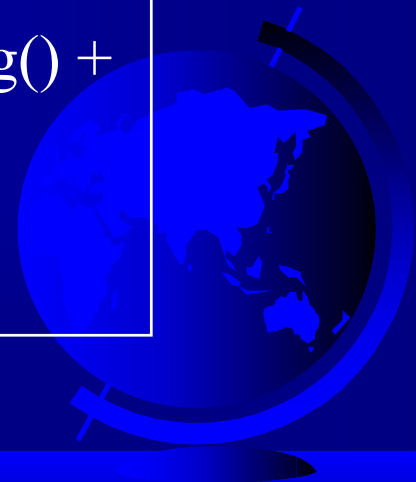
```
C:\WINNT\System32\cmd.exe

Please enter an integer:3
Please enter an integer:5
Please enter an integer:7
Please enter an integer:11
3/5 + 7/11 = 68/55
3/5 - 7/11 = -2/55
3/5 * 7/11 = 21/55
3/5 / 7/11 = 33/35
Press any key to continue . . . _
```

Java API and Core Java classes

java.lang

• Contains core Java classes, such as numeric classes, strings, and objects. This package is implicitly imported to every Java program.

• It includes Object, String, System, Math, Number, Character, Boolean, Byte, Short, Integer, Long, Float, and Double

Java API and Core Java classes

☞ `java.awt`

Contains classes for graphics, many components including window, frame, panel, menus, buttons, fonts, lists, and many others.

☞ `java.applet`

Contains classes for supporting applets.

Java API and Core Java classes, cont.

☞ `java.io`

Contains classes for input and output streams and files.

☞ `java.util`

Contains many utilities, such as date, calendar, locate, system properties, vectors, hashing, and stacks.

☞ `java.net`

Contains classes for supporting network communications.

Java API and Core Java classes, cont.

- javax.swing
  - Contains lightweight classes
- java.util
  - Contains many utilities, such as date, calendar, locate, system properties, vectors, hashing, and stacks.

- java.net
  - Contains classes for supporting network communications.

Java API and Core Java classes, cont.

☞ `java.awt.image`

Contains classes for managing bitmap images.

☞ `java.awt.peer`

Platform-specific GUI implementation.

☞ Others:

`java.sql`

`java.rmi`

The `String` Class

- ☞ Declaring a String:
  - – String message = "Welcome to Java!"
  - – String message = new String("Welcome to Java!");
  - – String s = new String();
- ☞ String Comparisons
- ☞ String Concatenation
- ☞ Substrings
- ☞ String Length
- ☞ Retrieving Individual Characters in a String

String Comparisons

equals

   String s1 = "Welcome";

   String s2 = "welcome";

                    Return boolean value

if (s1.equals(s2))

{ // s1 and s2 have the same contents }

if (s1 = = s2)

{

  // s1 and s2 have the same reference

}

String Comparisons

Use

  s1.compareTo(s2)

return 0 if s1 is equal to s2

less than 0 if s1 is lexicographically less than s2

greater than 0 if s1 is lexicographically greater than s2

Example:

s1="abc"   and   s2="abe"

  s1.compareTo(s2) return -2

String Concatenation

```
String s3 = s1.contact(s2);
```

String s3 = s1 + s2;

Concatenation

```
String s3 =s1.concat(s2);
```

concatenate s1 and s2 to s3

Alternatively, Use plus sign "+" to concatenate

```
String myString=message+" and " + " HTML";
```
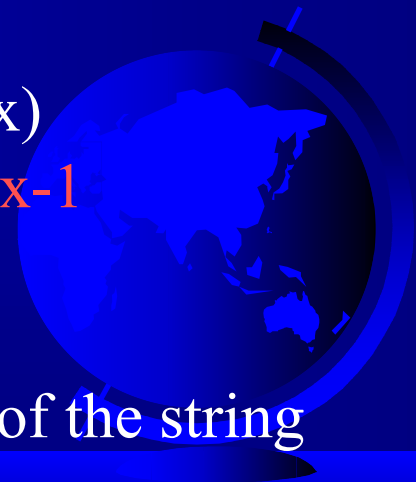
Substrings

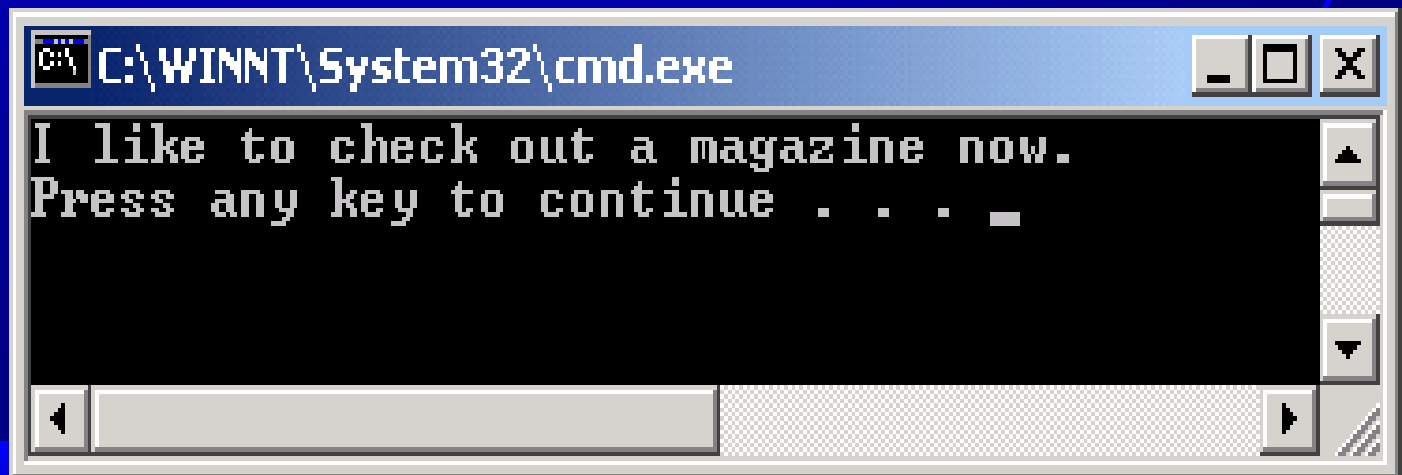`String` is an immutable class; its values cannot be changed individually.

```
String s1 = "Welcome to Java";
String s2 = s1.substring(0,10) + "HTML";
```

public String substring(int beginIndex, int endIndex)
    return a substring from beginIndex to endIndex-1

public String substring(int beginIndex)
    return a substring from beginIndex to the end of the string

```java
class CheckString
{
 public static void main(String[] args)
 {
 String statement="I like to check out a book now.";
 String st=statement.substring(0,22)
    +"magazine"+statement.substring(26);
 System.out.println(st);
 }
}
```

```
C:\WINNT\System32\cmd.exe
I like to check out a magazine now.
Press any key to continue . . . _
```

Finding String Length

Finding string length using the `length()` method:

```
message = "Welcome";
message.length()  (returns 7)
```

Retrieving Individual Characters in a String

☞ Do not use `message[0]`

☞ Use `message.charAt(index)`

☞ Index starts from `0`

Example 5.8
Finding Palindromes

☞Objective: Checking whether a string is a palindrome: a string that reads the same forward and backward.

```java
// CheckPalindrome.java: Check whether a string is a palindrome
public class CheckPalindrome
{
  // Main method
  public static void main(String[] args)
  {
    // Prompt the user to enter a string
    System.out.print("Enter a string: ");
    String s = MyInput.readString();
    if (isPalindrome(s))
    {
      System.out.println(s + " is a palindrome");
    }
    else
    {
      System.out.println(s + " is not a palindrome");
    }
  }
```

```java
// Check if a string is a palindrome
public static boolean isPalindrome(String s)
{
  // The index of the first character in the string
  int low = 0;
  // The index of the last character in the string
  int up = s.length() - 1;
  while (low < up)
  {
    if (s.charAt(low) != s.charAt(up))
      return false; // Not a palindrome

    low++;
    up--;
  }
  return true; // The string is a palindrome
 }
}
```

```
C:\WINNT\System32\cmd.exe

Enter a string: Toyota
Toyota is not a palindrome
Press any key to continue . . . _
```

```
C:\WINNT\System32\cmd.exe

Enter a string: atoyota
atoyota is a palindrome
Press any key to continue . . .
```

The `StringBuffer` Class

String class can not used to change the content, since the value of a string is fixed once the string is created.

The `StringBuffer` class is an alternative to the `String` class. In general, a string buffer can be used wherever a string is used.

`StringBuffer` is more flexible than `String`. You can add, insert, or append new contents into a string buffer.

StringBuffer Constructors

☞ `public StringBuffer()`
No characters, initial capacity 16 characters.

☞ `public StringBuffer(int length)`
No characters, initial capacity specified by the `length` argument.

☞ `public StringBuffer(String str)`
Represents the same sequence of characters as the `string` argument. Initial capacity 16 plus the length of the `string` argument.

Appending New Contents into a String Buffer

```
StringBuffer strBuf = new StringBuffer();
strBuf.append("Welcome");
strBuf.append(' ');
strBuf.append("to");
strBuf.append(' ');
strBuf.append("Java");
strBuf.insert(11,"HTML and ");
```

After position 10, or at 11, insert the literal string `"HTML and "`

More About String Buffer

```
public int capacity()
   return the current capacity of the
   string buffer
public synchronized StringBuffer reverse()
   reverses the sequence of the string
   comtained in the string buffer
public int lenth()
   return the number of characters in the
   string buffer
public synchronized charAt(int index)
   return the specific character at a
   specific index in the string buffer
```
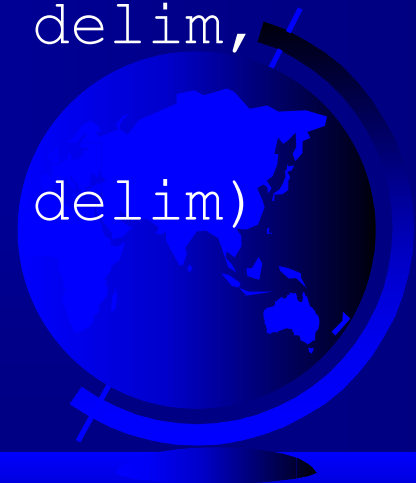
The `StringTokenizer` Class Constructors

☞ Break a string into pieces, used for coding and decoding

☞ Set up delimitters so we can break a string into pieces called tokens

☞ Default delimiters are " \t\n\r"

☞ StringTokenizer(String s, String delim, boolean returnTokens)

☞ StringTokenizer(String s, String delim)

☞ StringTokenizer(String s)

The `StringTokenizer` Class Methods

☞ `public boolean hasMoreTokens()`

☞ `public String nextToken()`

☞ `public String nextToken(String delim)`
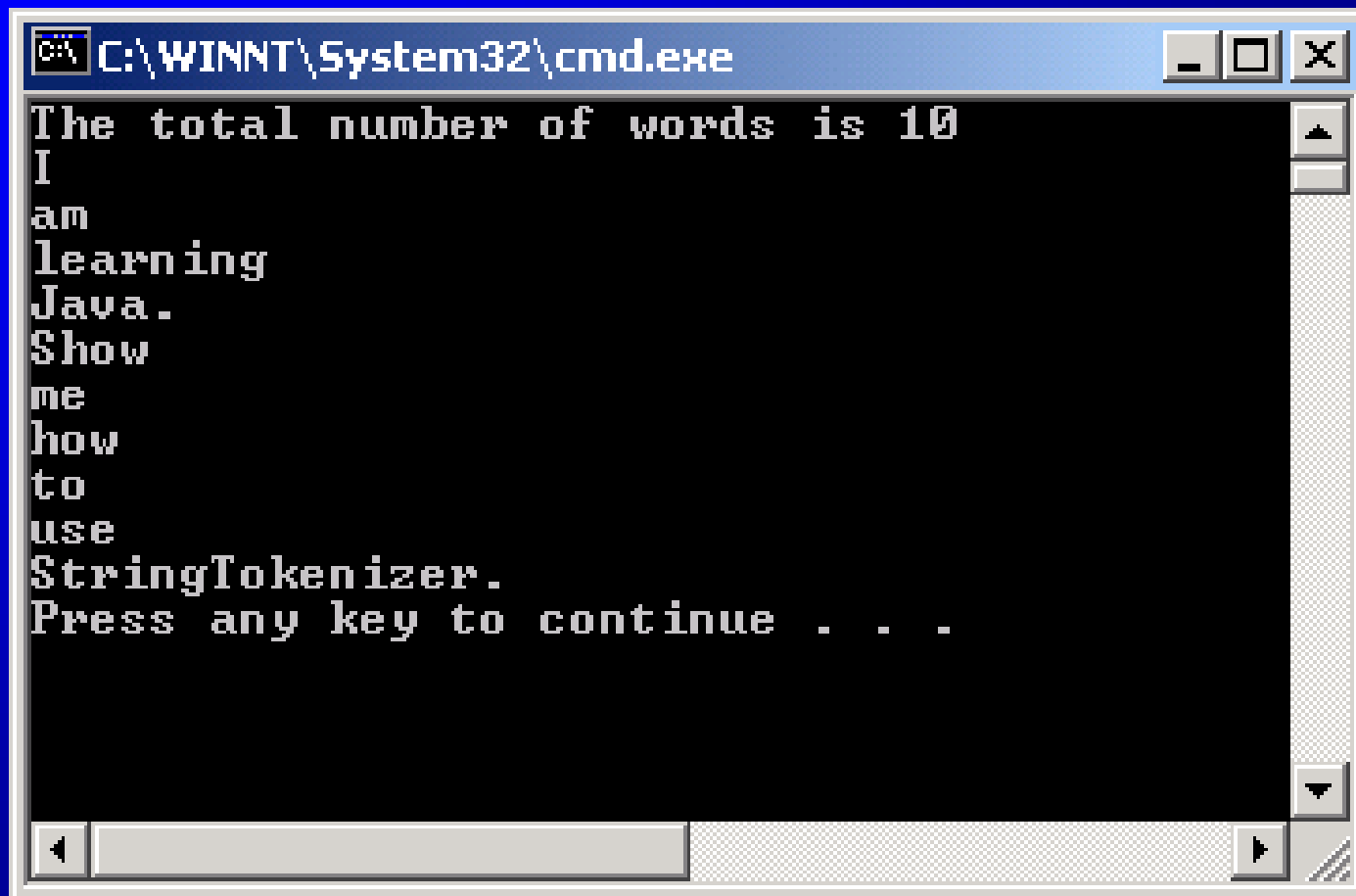
☞ `public String int countTokens()`

Example 5.10
Testing `StringTokenizer`

☞ Objective: Using a string tokenizer, retrieve words from a string and display them on the console.

```java
// TestStringTokenizer.java: Demonstrate StringTokenizer
import java.util.StringTokenizer;
public class TestStringTokenizer
{
  // Main method
  public static void main(String[] args)
  {
    // Create a string and string tokenizer
 String s = "I am learning Java. Show me how to use StringTokenizer.";
    StringTokenizer st = new StringTokenizer(s);
    // Retrieve and display tokens
    System.out.println("The total number of words is " +
      st.countTokens());
    while (st.hasMoreTokens())
      System.out.println(st.nextToken());
  }
}
```

```
C:\WINNT\System32\cmd.exe
The total number of words is 10
I
am
learning
Java.
Show
me
how
to
use
StringTokenizer.
Press any key to continue . . .
```