CISC 3115 TY3
C03a: Defining Class

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

- Review
- Defining object
- Defining class
- UML class diagram
- Constructors and default constructor
- Accessing objects via reference variables
- Primitive and reference variables
- Garbage collection

Review: What Have We Learned?

- Selections
- Iterations/Loops
- Methods
- Arrays

```
class HelloMsgs {
  public static void main(String[] args) {
    for (int i=0; i<10; i++) {
      if (i % 2 == 0) {
         System.out.println("Hello!");
      }
    }
  }
}
```

Recall: Authoring a Java Program

- Let’s consider the following 5 components
  - Requirement
  - Design
  - Implementation
  - Verification (commonly, testing)
  - Validation
- Call them 5 components instead of 5 steps, because it is not necessary to follow them in the above order

Recall: Requirements

- About answering question:
  - What does the “customer” want? Call the answer the requirement.
    - In the class:
      - What does the instructor want?
    - For your own exploration:
      - What do I want?
- Programmers provide a technical solutions in the means of software/programs to customers
- Is what we learned sufficient?

Object-Oriented Programming

- Programming using objects
- *An object* represents an entity in the real world that can be distinctly identified.
    - Student, instructor, class
    - Building, room, desk
    - Circle, rectangle
    - Button, menu
    - Loan, sales transaction

Object, State, and Behavior

- An object has a unique identity, state, and behaviors.
  - The *state* of an object consists of a set of *data fields* (also known as *properties*) with their current values.
  - The *behavior* of an object is defined by a set of methods representing what it does.

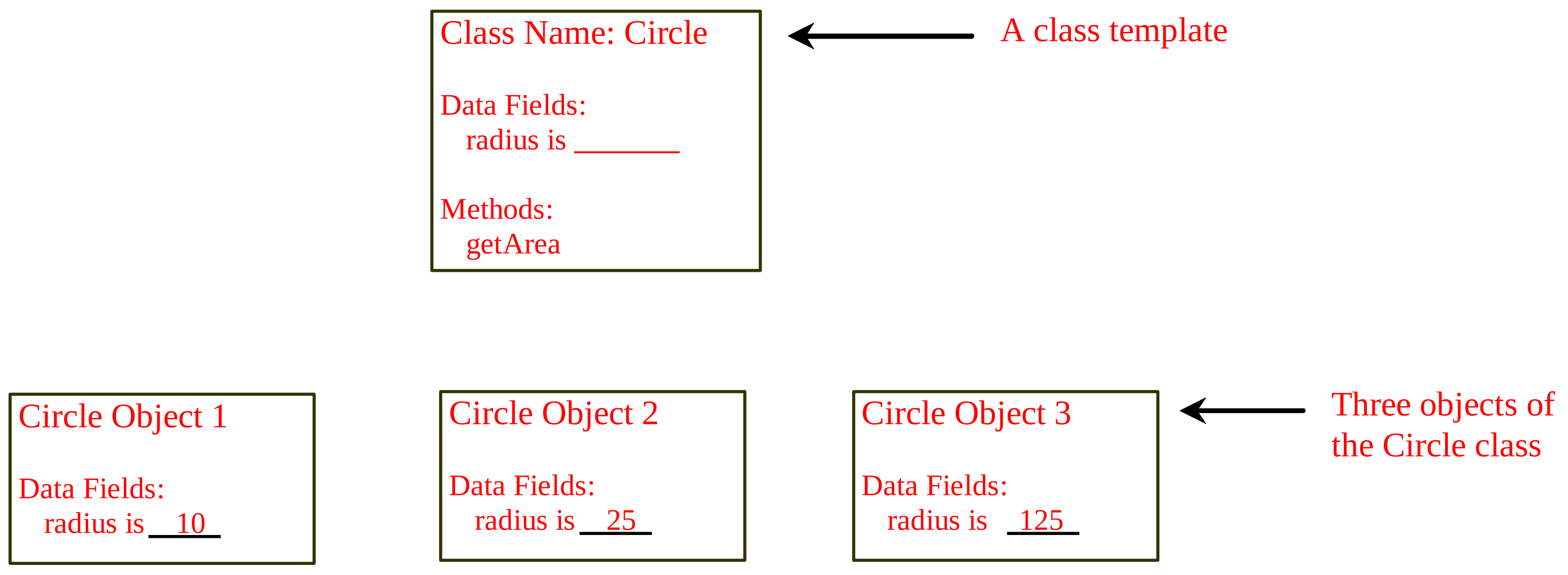
Classes

- Define objects of the same type, a template that an object can be created from.

- A Java class uses variables to define data fields and methods to define behaviors. Additionally, a class provides a special type of methods, known as constructors, which are invoked to construct objects from the class.

Objects and Classes

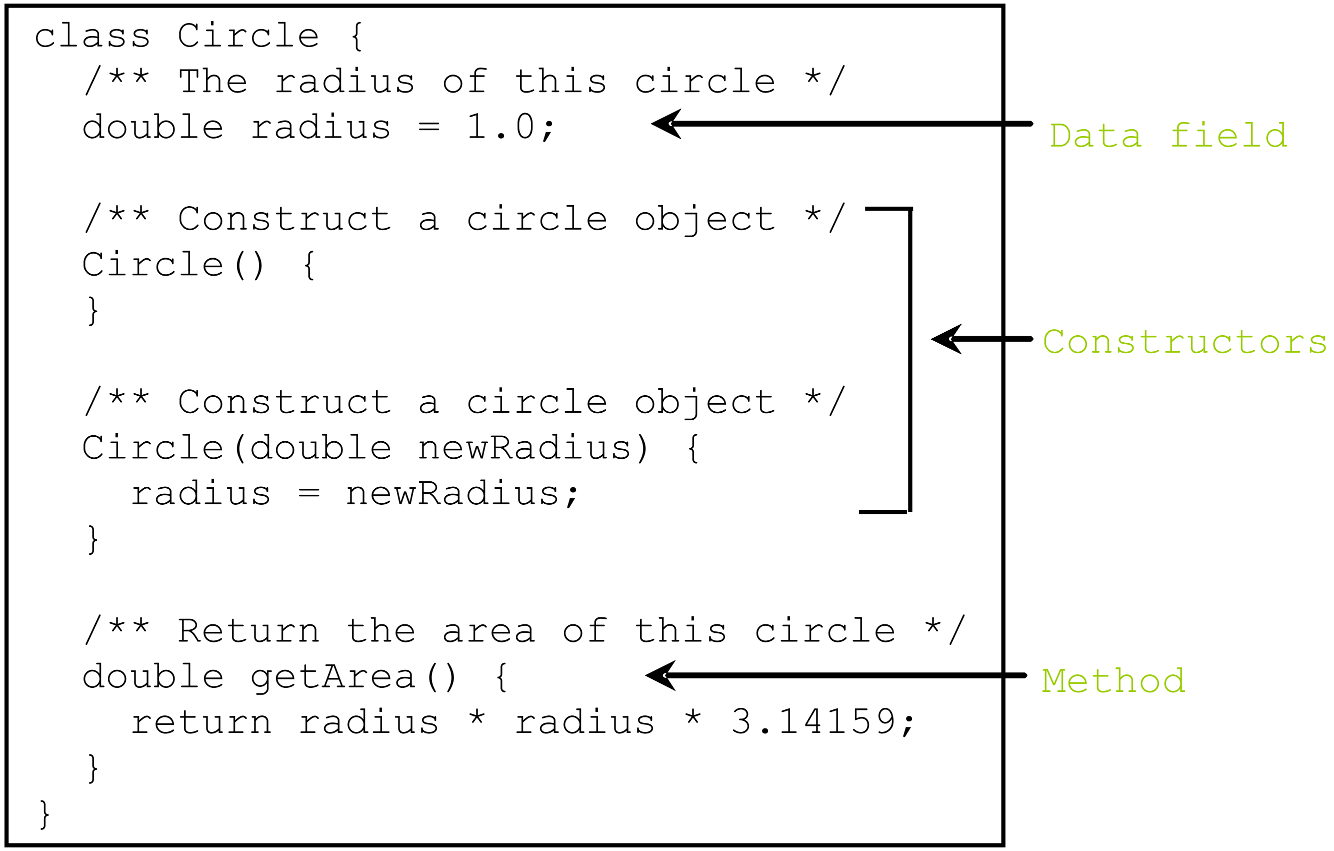
- From a class, we can create objects of the class



Objects and Classes: State and Behavior

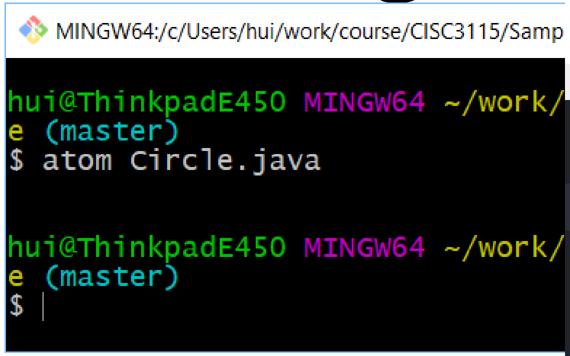
- A Java class uses variables to define data fields and methods to define behaviors.
  - The state of an object of the class corresponds to the data fields and their values.
  - The behavior of the object corresponds to the methods.
- Constructors
  - A special type of methods that are invoked to initialize the data fields when the object is being constructed.

A Circle Class



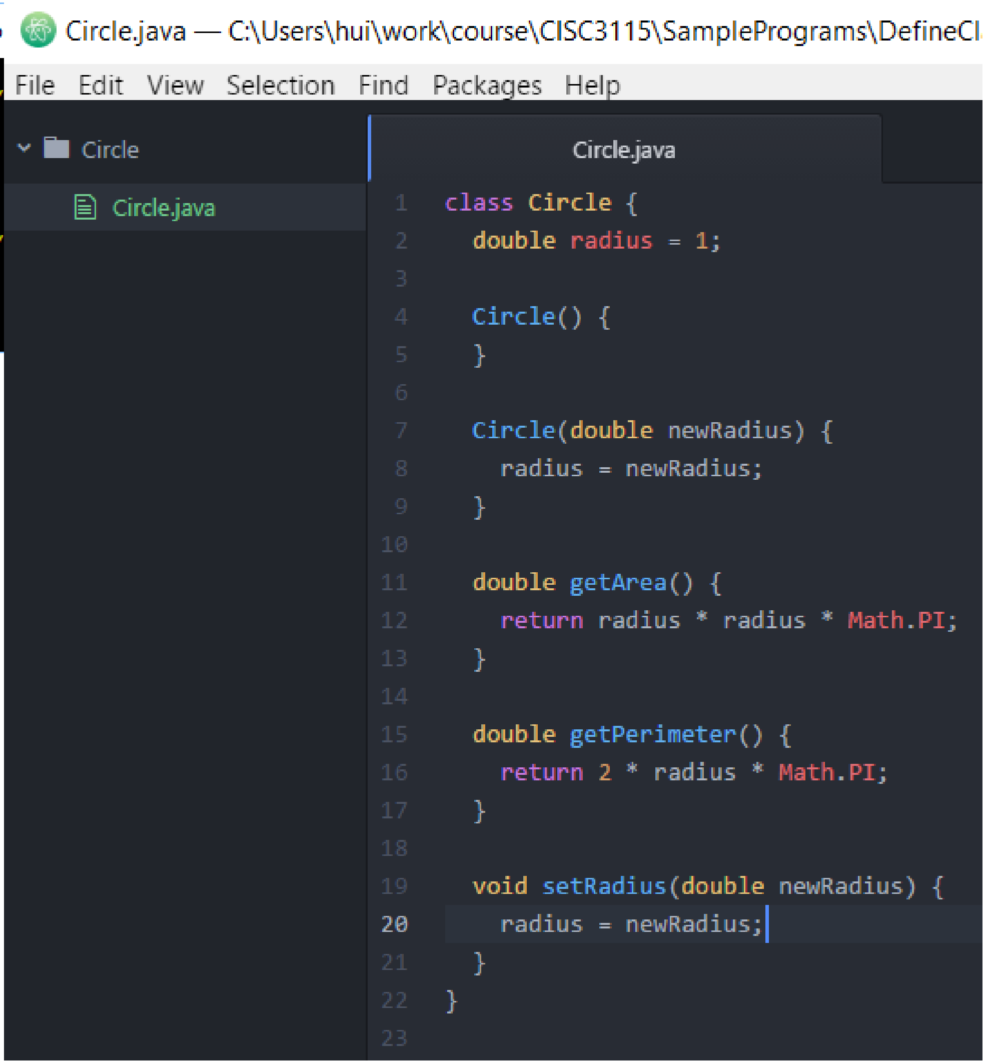
```
class Circle {
  /** The radius of this circle */
  double radius = 1.0;

  /** Construct a circle object */
  Circle() {
  }

  /** Construct a circle object */
  Circle(double newRadius) {
    radius = newRadius;
  }

  /** Return the area of this circle */
  double getArea() {
    return radius * radius * 3.14159;
  }
}
```

Writing the Circle Class



MINGW64:/c/Users/hui/work/course/CISC3115/Samp

```
hui@ThinkpadE450 MINGW64 ~/work/
e (master)
$ atom Circle.java

hui@ThinkpadE450 MINGW64 ~/work/
e (master)
$ 
```



Circle.java — C:\Users\hui\work\course\CISC3115\SamplePrograms\DefineCl
File Edit View Selection Find Packages Help
Circle
Circle.java
Circle.java

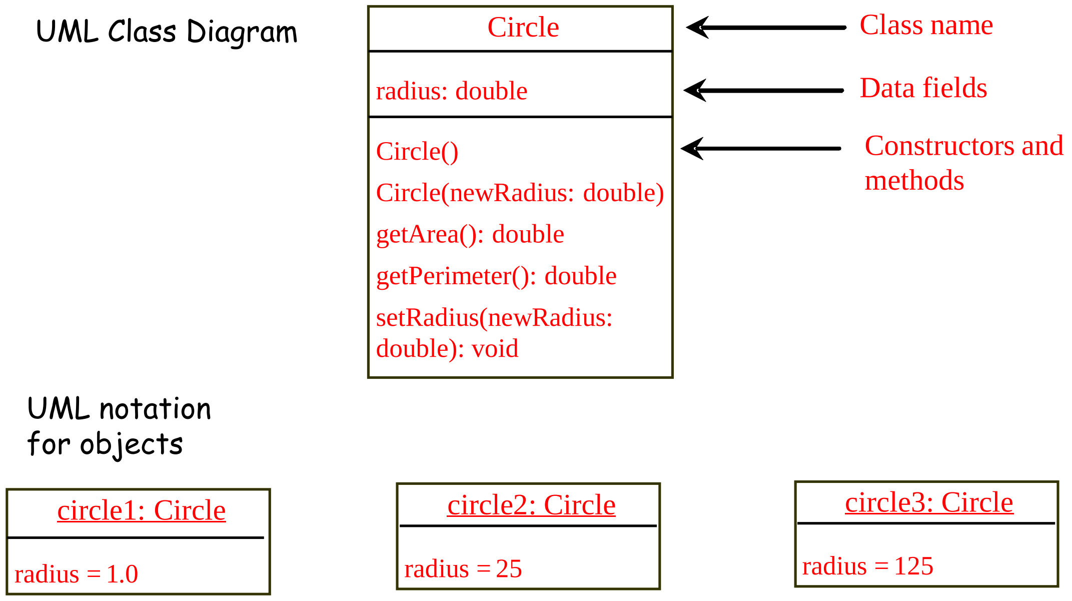
```java
class Circle {
  double radius = 1;

  Circle() {
  }

  Circle(double newRadius) {
    radius = newRadius;
  }

  double getArea() {
    return radius * radius * Math.PI;
  }

  double getPerimeter() {
    return 2 * radius * Math.PI;
  }

  void setRadius(double newRadius) {
    radius = newRadius;
  }
}
```

Reading the Circle Class

- Compared to the programs you written, is there any notable difference?
  - Does it have a main method?
  - Can you run it?
  - Can you compile it?
  - Is there any constructors? Where are they? How are constructors named? Does a constructor have a return type?
  - Where are the methods that define the behavior an object created from the class? Must a method have a return type? What are the return types?
  - Can method take a parameter? Must a parameter have a type and name?

Representing Class and Objects in UML Diagram

- UML = Unified Modeling Language



Reading the UML Diagram

- How does a class diagram depict a class?
  - How is a data field presented?
  - How is a constructor represented?
  - How is a method represented?
- How is an object represented in UML?

Questions

- Concepts of objects and classes
- Relationship between objects and classes
- Defining class in Java
- Depicting class in UML

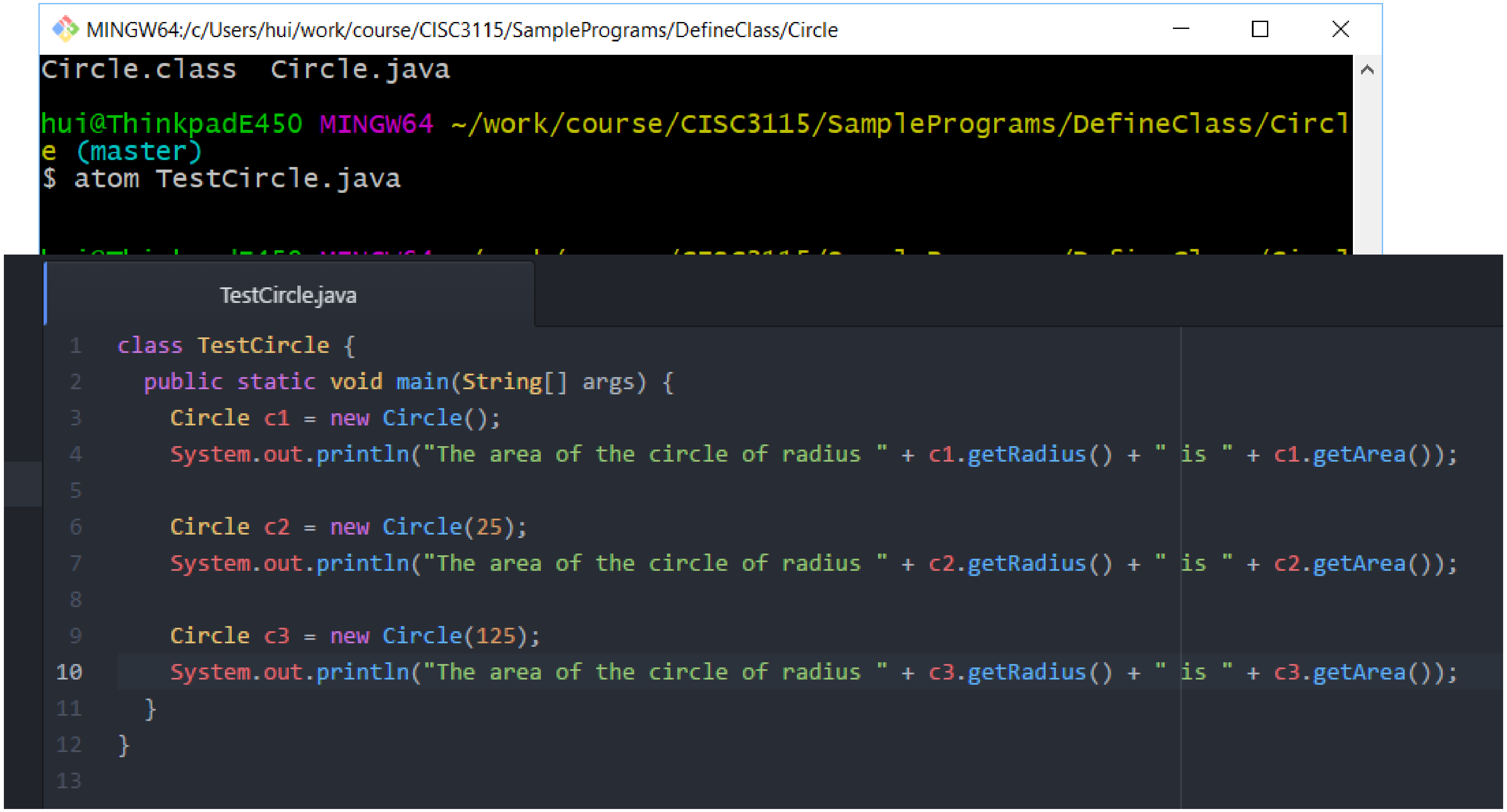
Observations

- We can compile the Circle class, but we cannot run it. Why can we not run it?
- The Circle class acts as a template from which objects of the Circle class can be created, but have we created any objects from the Circle class?

The TestCircle Class

- It has a main method
  - A number of Circle objects are created.
  - The areas of the Circle objects are computed and printed out

Writing the TestCircle Class

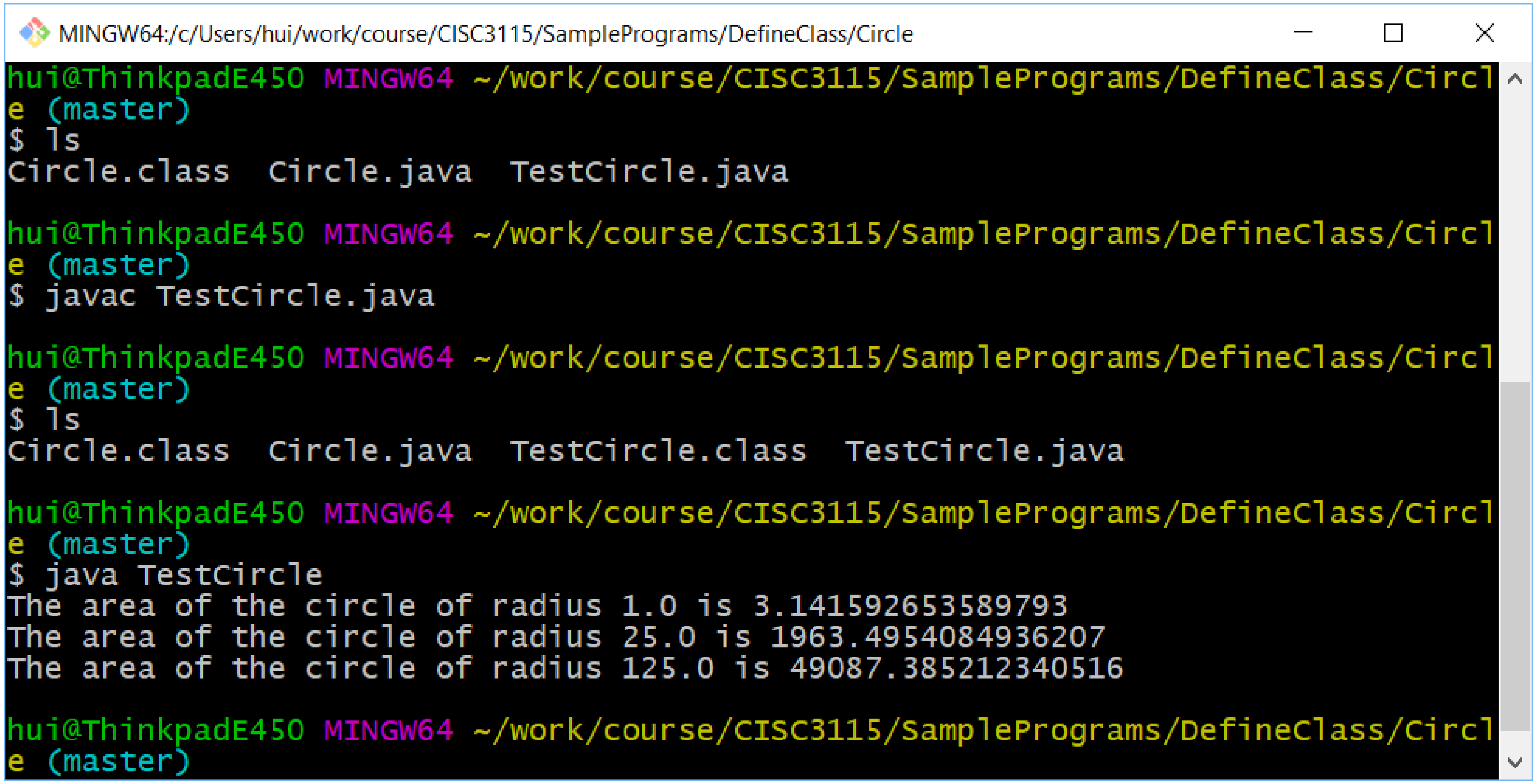

MINGW64:/c/Users/hui/work/course/CISC3115/SamplePrograms/DefineClass/Circle

```
Circle.class  Circle.java

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/SamplePrograms/DefineClass/Circl
e (master)
$ atom TestCircle.java
```

TestCircle.java

```java
class TestCircle {
  public static void main(String[] args) {
    Circle c1 = new Circle();
    System.out.println("The area of the circle of radius " + c1.getRadius() + " is " + c1.getArea());

    Circle c2 = new Circle(25);
    System.out.println("The area of the circle of radius " + c2.getRadius() + " is " + c2.getArea());

    Circle c3 = new Circle(125);
    System.out.println("The area of the circle of radius " + c3.getRadius() + " is " + c3.getArea());
  }
}
```

Compiling and Running the Program



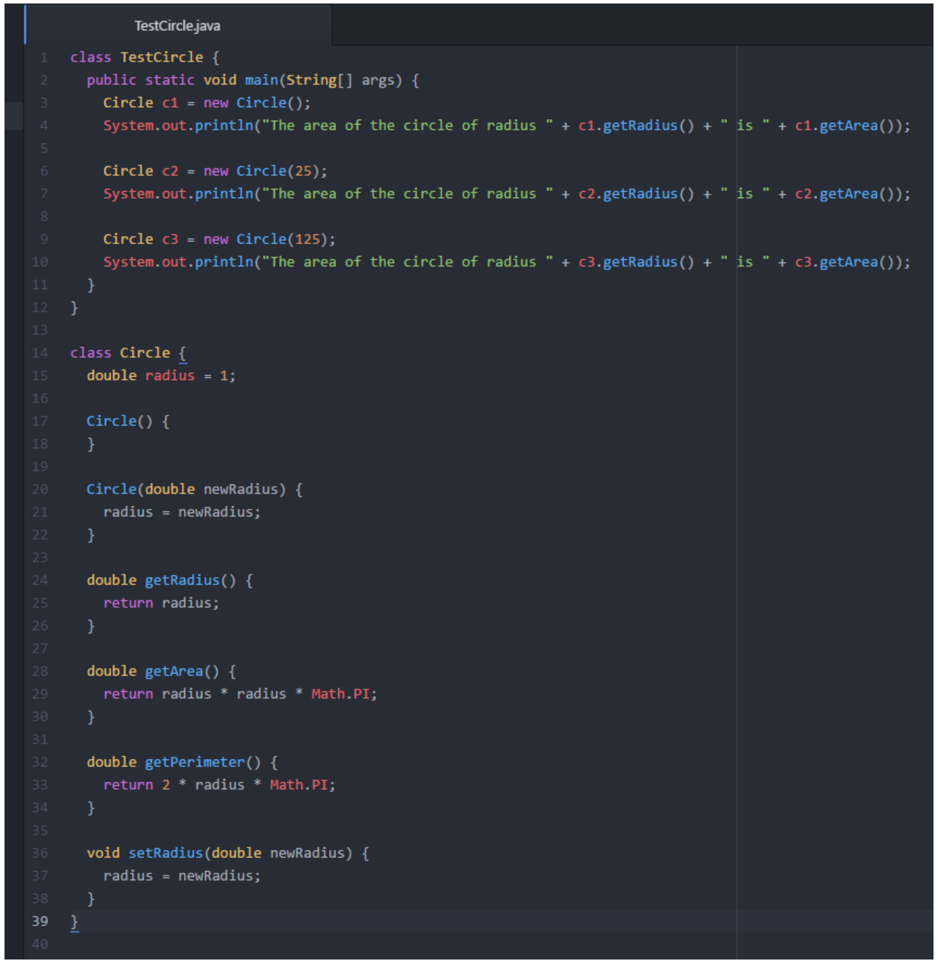
MINGW64:/c/Users/hui/work/course/CISC3115/SamplePrograms/DefineClass/Circle

```
hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/SamplePrograms/DefineClass/Circl
e (master)
$ ls
Circle.class  Circle.java  TestCircle.java

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/SamplePrograms/DefineClass/Circl
e (master)
$ javac TestCircle.java

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/SamplePrograms/DefineClass/Circl
e (master)
$ ls
Circle.class  Circle.java  TestCircle.class  TestCircle.java

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/SamplePrograms/DefineClass/Circl
e (master)
$ java TestCircle
The area of the circle of radius 1.0 is 3.141592653589793
The area of the circle of radius 25.0 is 1963.4954084936207
The area of the circle of radius 125.0 is 49087.385212340516

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/SamplePrograms/DefineClass/Circl
e (master)
```

Two Alternatively Methods to Write the Simple Program

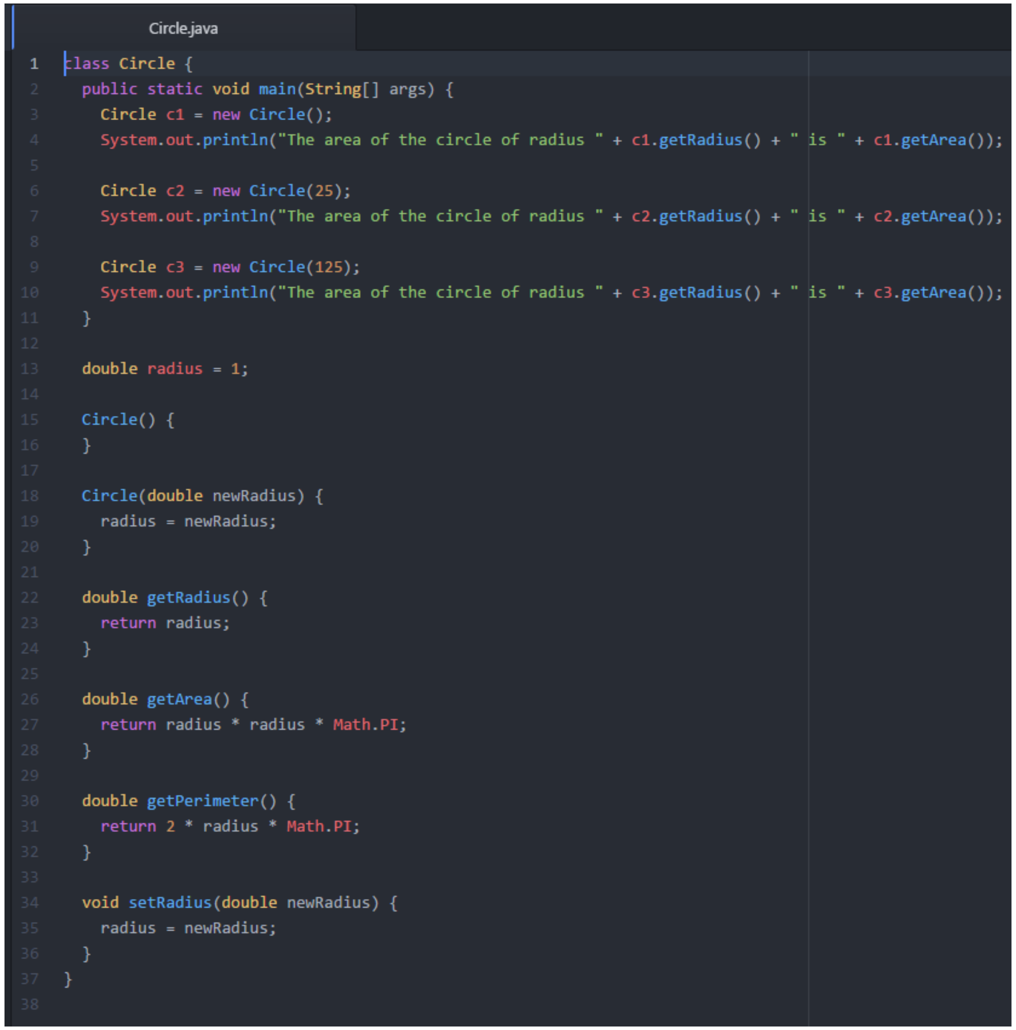
- One file has the two classes
- One file has the Circle class that has a main method

One File, Two Classes


TestCircle.java

```java
class TestCircle {
  public static void main(String[] args) {
    Circle c1 = new Circle();
    System.out.println("The area of the circle of radius " + c1.getRadius() + " is " + c1.getArea());

    Circle c2 = new Circle(25);
    System.out.println("The area of the circle of radius " + c2.getRadius() + " is " + c2.getArea());

    Circle c3 = new Circle(125);
    System.out.println("The area of the circle of radius " + c3.getRadius() + " is " + c3.getArea());
  }
}

class Circle {
  double radius = 1;

  Circle() {
  }

  Circle(double newRadius) {
    radius = newRadius;
  }

  double getRadius() {
    return radius;
  }

  double getArea() {
    return radius * radius * Math.PI;
  }

  double getPerimeter() {
    return 2 * radius * Math.PI;
  }

  void setRadius(double newRadius) {
    radius = newRadius;
  }
}
```

One Class with Main Method



Circle.java

```java
class Circle {
  public static void main(String[] args) {
    Circle c1 = new Circle();
    System.out.println("The area of the circle of radius " + c1.getRadius() + " is " + c1.getArea());

    Circle c2 = new Circle(25);
    System.out.println("The area of the circle of radius " + c2.getRadius() + " is " + c2.getArea());

    Circle c3 = new Circle(125);
    System.out.println("The area of the circle of radius " + c3.getRadius() + " is " + c3.getArea());
  }

  double radius = 1;

  Circle() {
  }

  Circle(double newRadius) {
    radius = newRadius;
  }

  double getRadius() {
    return radius;
  }

  double getArea() {
    return radius * radius * Math.PI;
  }

  double getPerimeter() {
    return 2 * radius * Math.PI;
  }

  void setRadius(double newRadius) {
    radius = newRadius;
  }
}
```

Questions?

- Defining classes and creating objects

In-Class Exercise C03a-1

- Write two classes, TV and TestTV as illustrated in Listings 9.3 and 9.4 in the textbook
- Compile and run the program
- Make a submission
  - In your weekly practice repository, create a C03a-1 directory (assuming you have completed C02c-1 and know where your repository is on your computer)
  - Copy the programs you wrote to the directory
  - Use git to make a submission,
    - In the C03a-1 directory, do
      - git add TV.java TestTV.java
      - git commit –m "your message"
      - git push