

JAVA PROGRAMMING COURSE

EXERCISE

ABSTRACT CLASSES IN JAVA

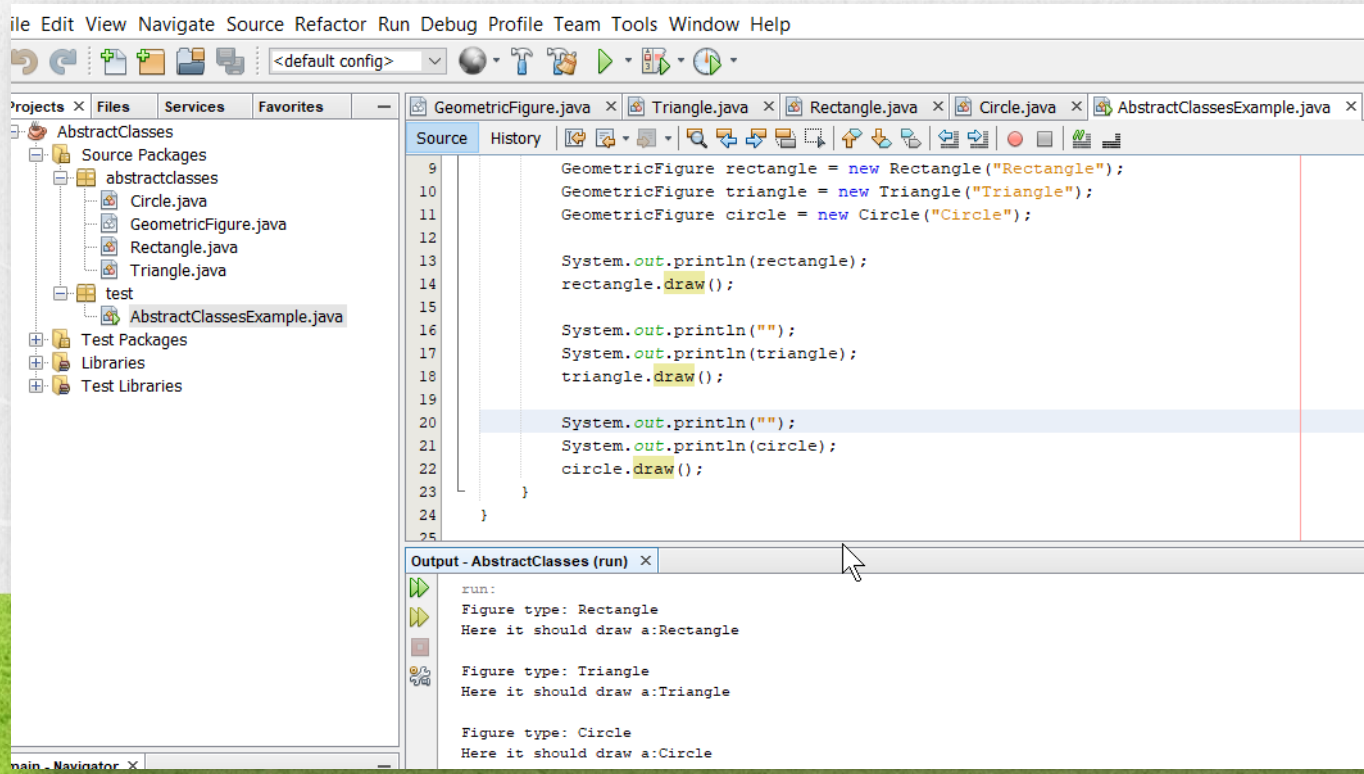


JAVA PROGRAMMING COURSE

www.globalmentoring.com.mx

EXERCISE OBJECTIVE

Create a program to practice the concept of Abstract Classes in Java. At the end we should observe the following:



The screenshot shows an IDE with the following components:

- Project Explorer:** Shows a project named 'AbstractClasses' with a package 'abstractclasses' containing 'Circle.java', 'GeometricFigure.java', 'Rectangle.java', and 'Triangle.java'. There is also a 'test' package with 'AbstractClassesExample.java'.
- Source Editor:** Displays the code for 'GeometricFigure.java' (lines 9-25). The code creates instances of 'Rectangle', 'Triangle', and 'Circle' and calls their 'draw()' methods.
- Output Console:** Shows the output of the program, which prints the type of each object and a message indicating it should draw.

```
9      GeometricFigure rectangle = new Rectangle("Rectangle");
10     GeometricFigure triangle = new Triangle("Triangle");
11     GeometricFigure circle = new Circle("Circle");
12
13     System.out.println(rectangle);
14     rectangle.draw();
15
16     System.out.println("");
17     System.out.println(triangle);
18     triangle.draw();
19
20     System.out.println("");
21     System.out.println(circle);
22     circle.draw();
23 }
24
25 }
```

Output - AbstractClasses (run) X

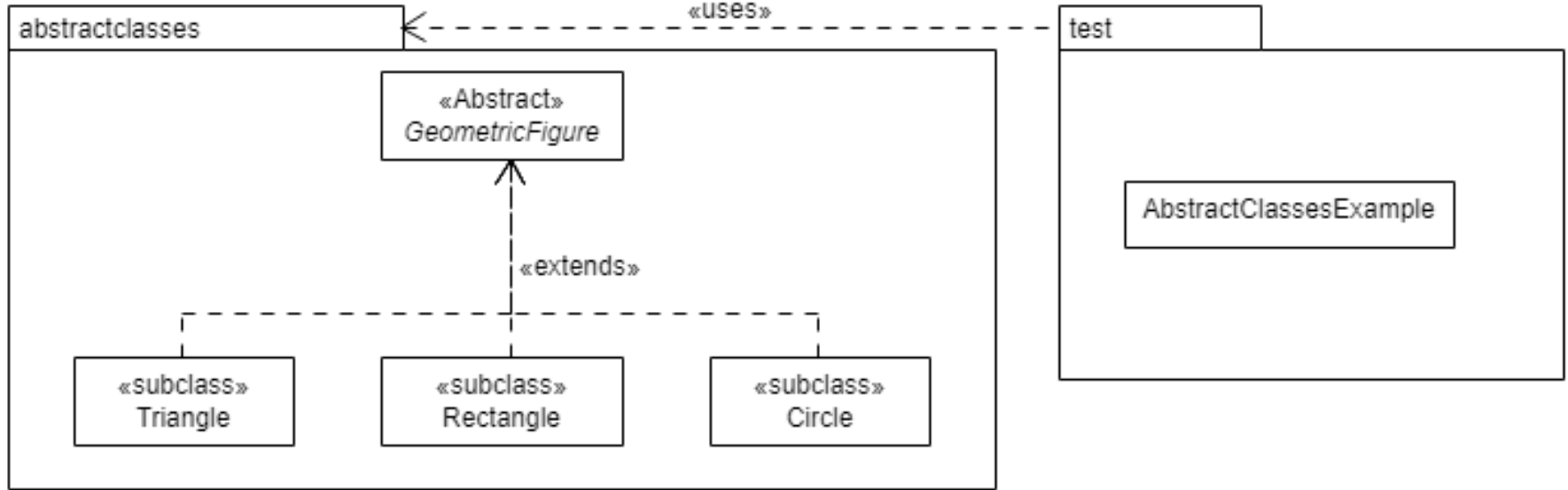
```
run:
Figure type: Rectangle
Here it should draw a:Rectangle

Figure type: Triangle
Here it should draw a:Triangle

Figure type: Circle
Here it should draw a:Circle
```

UML DIAGRAM

The project will be based on the following class diagram:



1. CREATE A NEW PROJECT

Create a new Project:

New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: AbstractClasses

Project Location: C:\Courses\JavaProgramming\Lesson14 Browse...

Project Folder: C:\Courses\JavaProgramming\Lesson14\AbstractClasses

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder: Browse...

Different users and projects can share the same compilation libraries (see Help for details).

☐ Create Main Class abstractclasses.AbstractClasses

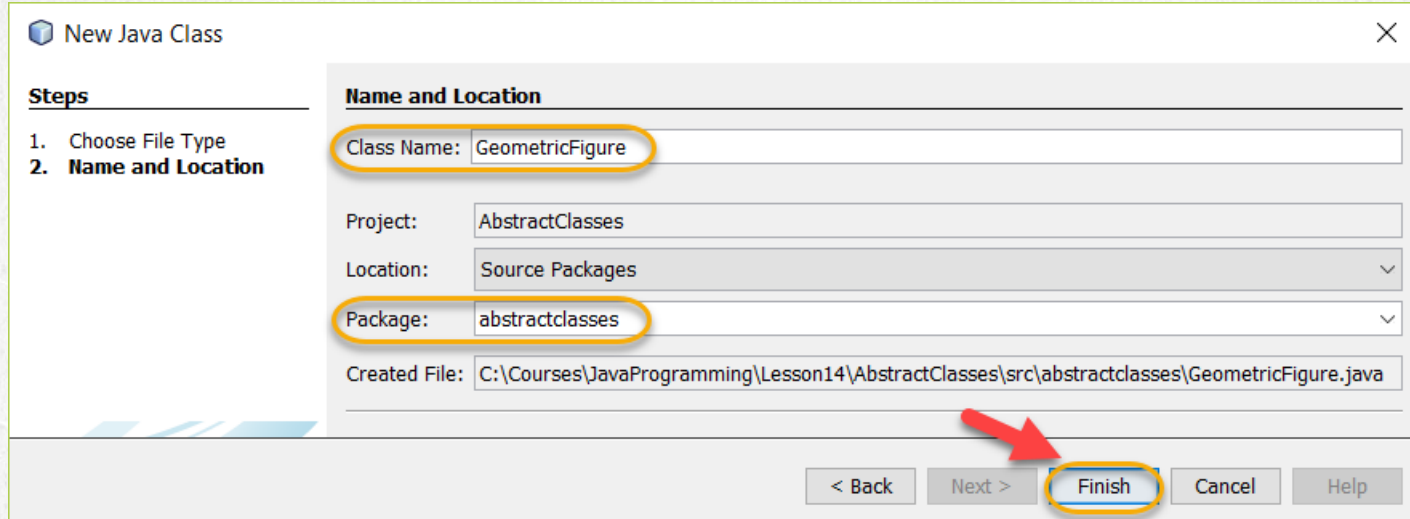
< Back Next > **Finish** Cancel Help

JAVA PROGRAMMING COURSE

www.globalmentoring.com.mx

2. CREATE A NEW CLASS

Create a new class:



New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

Package:

Created File:

< Back Next > **Finish** Cancel Help

3. MODIFY THE CODE

GeometricFigure.java:

```
package abstractclasses;

public abstract class GeometricFigure {

    protected String figureType;

    protected GeometricFigure(String figureType){
        this.figureType = figureType;
    }

    //The parent class does not define behavior
    public abstract void draw();

    public String getFigureType() {
        return figureType;
    }

    public void setFigureType(String figureType) {
        this.figureType = figureType;
    }

    @Override
    public String toString() {
        return "Figure type: " + this.figureType;
    }
}
```

4. CREATE A NEW CLASS

Create a new class:

New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

Package:

Created File:

< Back Next > **Finish** Cancel Help

5. MODIFY THE CODE

Triangle.java:

```
package abstractclasses;

public class Triangle extends GeometricFigure{

    public Triangle(String figureType) {
        super(figureType);
    }

    @Override
    public void draw() {
        //Implementation of the inherit drawing method of the GeometricFigure class
        System.out.println("Here it should draw a:" + this.getClass().getSimpleName());
    }
}
```


6. CREATE A NEW CLASS

Create a new class:

New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

Package:

Created File:

< Back Next > **Finish** Cancel Help

JAVA PROGRAMMING COURSE

www.globalmentoring.com.mx

7. MODIFY THE CODE

Rectangle.java:

```
package abstractclasses;

public class Rectangle extends GeometricFigure{

    public Rectangle(String figureType) {
        super(figureType);
    }

    @Override
    public void draw() {
        //Implementation of the inherit drawing method of the GeometricFigure class
        System.out.println("Here it should draw a:" + this.getClass().getSimpleName());
    }
}
```

8. CREATE A NEW CLASS

Create a new class:

New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

Package:

Created File:

< Back Next > **Finish** Cancel Help

JAVA PROGRAMMING COURSE

www.globalmentoring.com.mx

9. MODIFY THE CODE

Circle.java:

```
package abstractclasses;

public class Circle extends GeometricFigure{

    public Circle(String figureType) {
        super(figureType);
    }

    @Override
    public void draw() {
        //Implementation of the inherit drawing method of the GeometricFigure class
        System.out.println("Here it should draw a:" + this.getClass().getSimpleName());
    }
}
```


10. CREATE A NEW CLASS

Create a new class:

New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

Package:

Created File:

< Back Next > **Finish** Cancel Help

11. MODIFY THE CODE

AbstractClassExample.java:

```
package test;

import abstractclasses.*;

public class AbstractClassesExample {

    public static void main(String args[]) {
        //Creation of objects
        GeometricFigure rectangle = new Rectangle("Rectangle");
        GeometricFigure triangle = new Triangle("Triangle");
        GeometricFigure circle = new Circle("Circle");

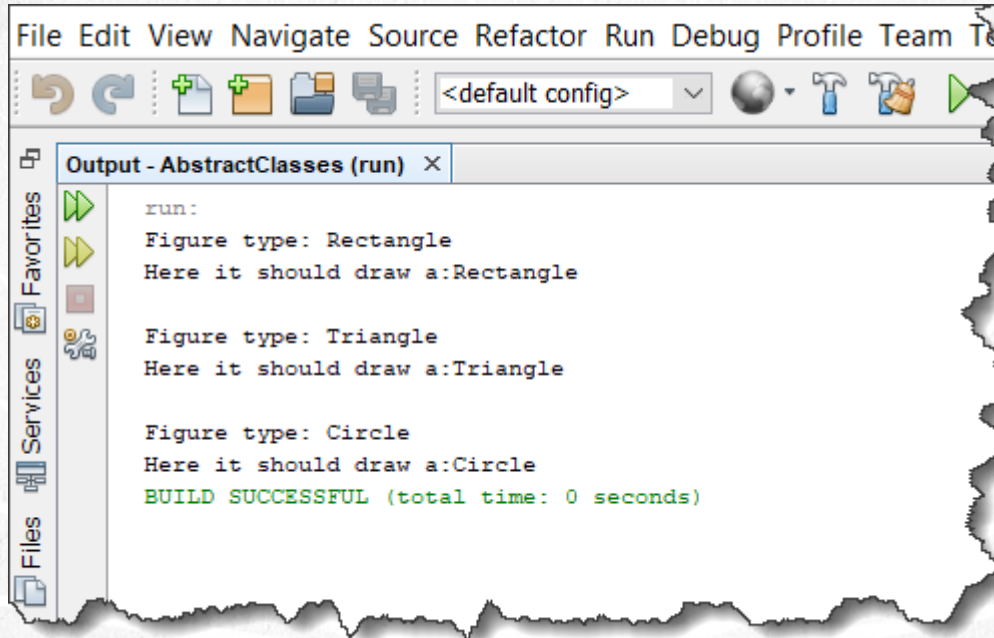
        System.out.println(rectangle);
        rectangle.draw();

        System.out.println("");
        System.out.println(triangle);
        triangle.draw();

        System.out.println("");
        System.out.println(circle);
        circle.draw();
    }
}
```

12. EXECUTE THE PROJECT

The result is as follows:

A screenshot of an IDE's Run Output window. The window title is "Output - AbstractClasses (run)". The output text shows the execution of a Java program that draws shapes. It starts with "run:", followed by "Figure type: Rectangle" and "Here it should draw a:Rectangle". Then it shows "Figure type: Triangle" and "Here it should draw a:Triangle". Next is "Figure type: Circle" and "Here it should draw a:Circle". Finally, it ends with "BUILD SUCCESSFUL (total time: 0 seconds)". The IDE's menu bar (File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team) and a toolbar with various icons are visible at the top of the window. On the left side, there is a sidebar with icons for Favorites, Services, and Files.

```
run:
Figure type: Rectangle
Here it should draw a:Rectangle

Figure type: Triangle
Here it should draw a:Triangle

Figure type: Circle
Here it should draw a:Circle
BUILD SUCCESSFUL (total time: 0 seconds)
```

JAVA PROGRAMMING COURSE

www.globalmentoring.com.mx

EXERCISE CONCLUSION

- With this exercise we have put into practice the concept of abstract classes in Java.
- We have seen how the same polymorphism rules apply, however, an abstract class can not be instantiated, and therefore a subclass is required to implement the abstract method defined in the parent class.



JAVA PROGRAMMING COURSE

www.globalmentoring.com.mx

ONLINE COURSE

JAVA PROGRAMMING

By: Eng. Ubaldo Acosta



JAVA PROGRAMMING COURSE

www.globalmentoring.com.mx