INSTITUTE OF TECHNOLOGY TRALEE

WINTER EXAMINATIONS AY 2015-2016

Object Oriented Programming 2

Module Code PROG61004

CRN 43840

External Examiner: Ms Sabrina Spillane

Internal Examiner: Mr John Walsh

Duration: 2 Hours

Instructions to Candidates: Please answer **question 1** and any **two other questions.**

Question 1

a) Write a brief note explaining the OOP principle of **Encapsulation**. Include an explanation of **true encapsulation** in your answer. (6 Marks)

b) Write the class definition for a *Boat.* The *Boat* should have the following three **attributes**: boatName: String, boatLength:double and isSailBoat: boolean. The class should have appropriate **mutator** and **accessor** methods defined for all attributes as well as a **toString** method that uses the **String format method** to prepare its output. The class should have a **no-argument constructor** that sets up the attributes indirectly via a call to a second **3-argument constructor**. (12 Marks)

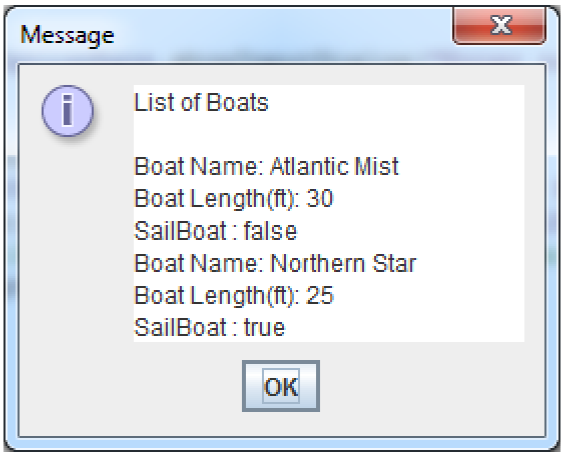
c) Java provides a powerful tool called **javadoc** for generating web pages to describe classes. Write javadoc style comments, including a general **description**, **@param** and **@return** tags for one accessor and one mutator method defined in part b) above.

(4 Marks)

d) Write a minimal **driver** class called **BoatClub**, including any import statements, that creates an **Array** of boats called *fleet*. The size of the array and the values for each attribute of a boat should be requested from the user using **input dialog boxes**.

(8 Marks)

e) Write an additional **class** method called displayBoats for inclusion in the driver class written for part d) that displays the Array of Boats. The statement calling this method from the main method is **displayBoats(fleet);**. The output is displayed in a message dialog box containing a **JTextArea**. An example output is shown below.



```
List of Boats

Boat Name: Atlantic Mist
Boat Length(ft): 30
SailBoat : false
Boat Name: Northern Star
Boat Length(ft): 25
SailBoat : true
```

(10 Marks)

Question 2

A Fraction class is being developed to facilitate arithmetic with fractions. A fractional number has the form: **numerator/denominator** where numerator:int and denominator:int are **attributes** of the class. You can assume that the code for the attributes and standard accessor and mutator methods is written.

The following three lines were included in a driver class that tests the functionality of the Fraction class, where fa,fb,fc,f1,f2,f3 are objects of the Fraction class.

fa = f1.add(f2);

fb= Fraction.add(f1,f2);

fc= Fraction.add(f1,f2,f3);

a) Write a brief note explaining the OOP principle of **Abstraction.** (6 Marks)

b) Write the code for the add method of the Fraction class that will allow the statement *fa = f1.add(f2);* to execute. (9 Marks)

c) Using the method written for part b) write the code for the add method of the Fraction class that will allow the statement *fb= Fraction.add(f1,f2);* to execute. (5 Marks)

d) Using the method written for part b) write the code for the add method of the Fraction class that will allow the statement *fc= Fraction.add(f1,f2,f3);* to execute. (6 Marks)

e) Explain the OOP concepts of method **Overloading**. Give an example as part of your explanation. (4 Marks)

Question 3

The class BoatClubApp.java (**Appendix 1**) is a **JFrame** based **GUI** program that allows the user to set up and maintain a **collection** of boats for a boat club. Each boat is an object of the Boat class developed in Question1. The collection is held in a **LinkedList** of Boat objects called boats.

a) Explain the OOP concept(s) of **inheritance**. You should refer to the implementation of these concepts in the **BoatClubApp** system, in particular the **class header**, the **keywords** used and specific **methods** included in the **constructor** and **required elsewhere** in the program. (10 Marks)

b) Write code for the 'open' method, suitable for inclusion in BoatClubApp.java, that will **read** the **entire collection** of boats from a binary file called 'boats.dat'. See Appendix 2 for some useful information for this part.
(6 Marks)

c) Write a note on how **exceptions** are handled in Java. Include **examples** of types of exceptions that can occur.
(8 Marks)

d) Rewrite the code for part b) such that any IOException or FileNotFoundException is dealt with individually by the 'open' method.
(6 Marks)

Question 4

The class BoatClubApp.java **(Appendix 1)** is a **JFrame** based **GUI** program that allows the user to set up and maintain a **collection** of boats for a boat club. The menu system has two menus, one called 'File' that allows for creating, opening and saving a file as well as quitting the program, the other called 'Boat' that allows for adding a new boat and displaying the entire collection of boats. Examine the code and answer the following questions.

a) Explain what you understand by the **scope** of a variable, in particular discussing the scope of the **LinkedList boats** variable and the **JMenuItem item** variables in the BoatClubApp class. (7 Marks)

b) Write code for the 'display' method suitable for inclusion in BoatClubApp.java. The method should use a **JTextArea** object to display the details of all the boats using an **iterator** to scan through the **LinkedList** to retrieve the appropriate details for each Boat object. (10 Marks)

c) Explain the use of the keyword *this* in the createFileMenu method, in particular referring to its use in dealing with events. (6 Marks)

d) Write the partial code, including the **header**, for the **appropriate method** that **invokes** the ‘addBoat’ and ‘display’ methods when the appropriate event is detected by the program. There is no requirement to write code to handle any of the items in the ‘File’ menu. See Appendix 2 for some useful information for this part. (7 Marks)

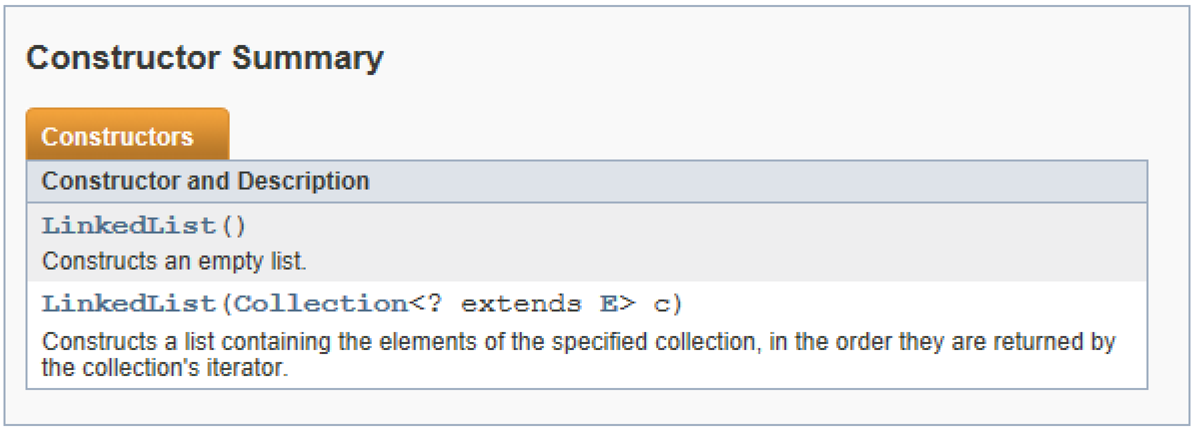
Appendix 1 BoatClubApp.java

```
import java.awt.event.*;
import java.awt.*;
import java.io.*;
import java.util.*;
import javax.swing.*;
// manages a collection of boats, holding the info in an LinkedList
public class BoatClubApp extends JFrame implements ActionListener{
                JMenu fileMenu,boatClubMenu;
                LinkedList <Boat> boats;

public static void main( String[] args ) {
               BoatClubApp frame = new BoatClubApp();
               frame.setVisible(true);}

public BoatClubApp( ) {
               boats = new LinkedList<Boat>();
               setTitle     ( "BoatClubApp" );
               setSize      ( 400,200 );
               setLocation  ( 100,100 );
               Container pane = getContentPane();
               setDefaultCloseOperation( EXIT_ON_CLOSE );
               createFileMenu();
               createBoatClubMenu();
               JMenuBar menuBar = new JMenuBar();
               setJMenuBar(menuBar);
               menuBar.add(fileMenu);
               menuBar.add(boatClubMenu);
   } // end
  public void newSystem() {
               boats = new LinkedList<Boat>();
   } // end new system

   public void save(){
               //code not given
   } //end save

   public void open() {
               //code not given
    } // end open()

//method to add a boat

    } // end add boat method

 // method to display all boats

     } // end display

 private void createFileMenu(){
        // create the menu
               fileMenu = new JMenu("File");
               // declare a menu item (re-usable)
               JMenuItem item;
               item = new JMenuItem("Save");
```

```
            item.addActionListener(this);
            fileMenu.add(item);
            item = new JMenuItem("Open");
            item.addActionListener(this);
            fileMenu.add(item);
            item = new JMenuItem("New File");
            item.addActionListener(this);
            fileMenu.add(item);
            // create the 'quit' option
            fileMenu.addSeparator();
            item = new JMenuItem("Quit");
            item.addActionListener(this);
            fileMenu.add(item);
    }

 private void createBoatClubMenu(){
            // create the menu
            boatClubMenu = new JMenu("Boat");
            // declare a menu item (re-usable)
            JMenuItem item;

            item = new JMenuItem("Add");
            item.addActionListener(this);
            boatClubMenu.add(item);

            item = new JMenuItem("Display");
            item.addActionListener(this);
            boatClubMenu.add(item);
    }


// event handling code here

}
```

Appendix 2: Extracts from the Java API

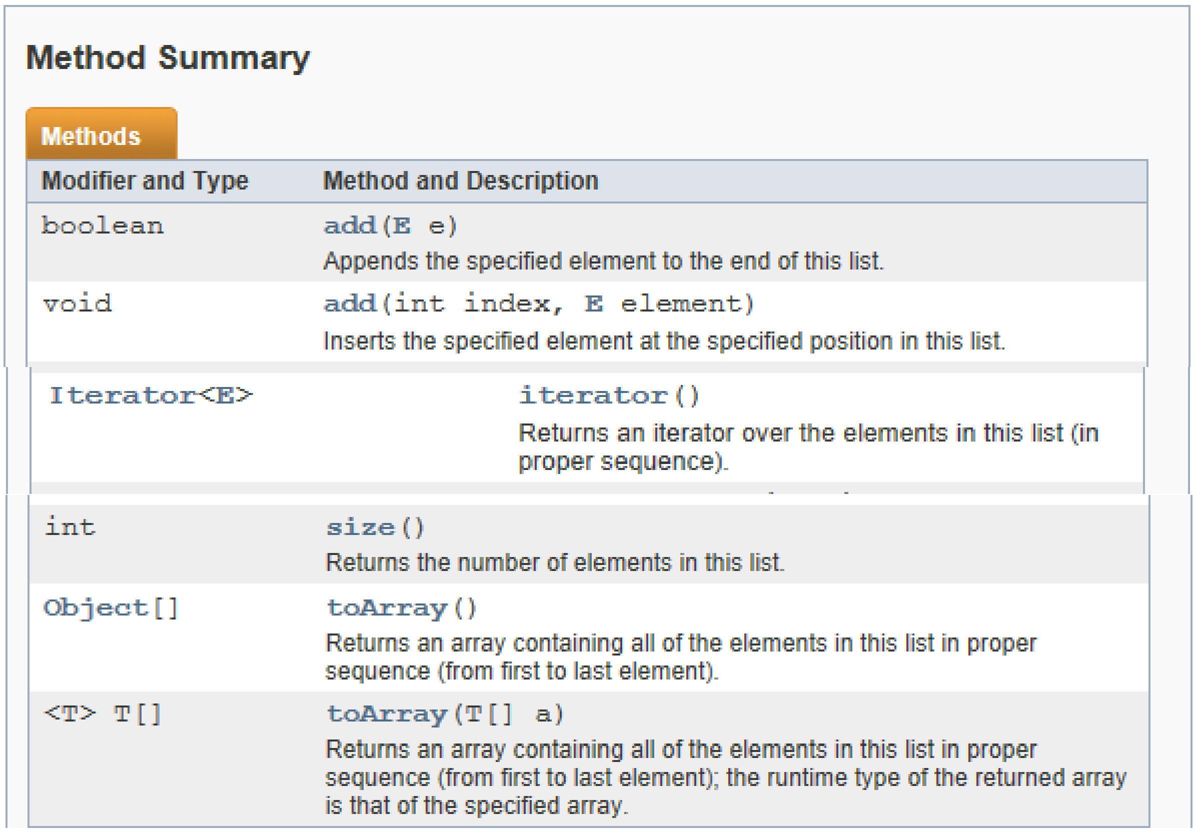
Class LinkedList<E>



Constructor Summary

Constructors

Constructor and Description
`LinkedList()` Constructs an empty list.
`LinkedList(Collection<? extends E> c)` Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.

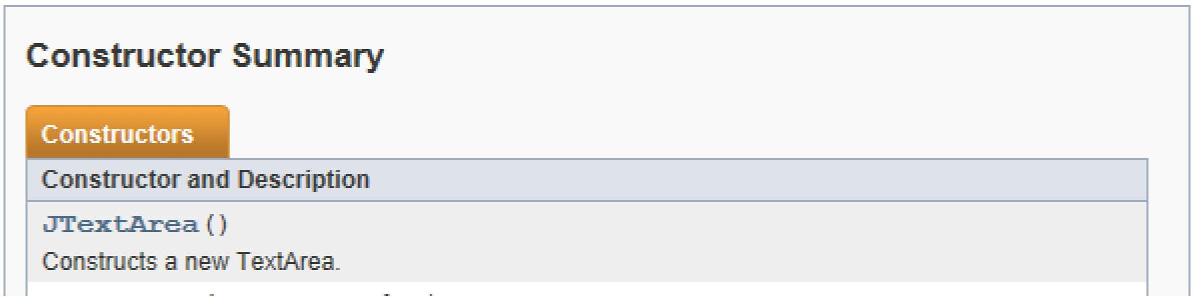


Method Summary

Methods

Modifier and Type	Method and Description
`boolean`	`add(E e)` Appends the specified element to the end of this list.
`void`	`add(int index, E element)` Inserts the specified element at the specified position in this list.
`Iterator<E>`	`iterator()` Returns an iterator over the elements in this list (in proper sequence).
`int`	`size()` Returns the number of elements in this list.
`Object[]`	`toArray()` Returns an array containing all of the elements in this list in proper sequence (from first to last element).
`<T> T[]`	`toArray(T[] a)` Returns an array containing all of the elements in this list in proper sequence (from first to last element); the runtime type of the returned array is that of the specified array.

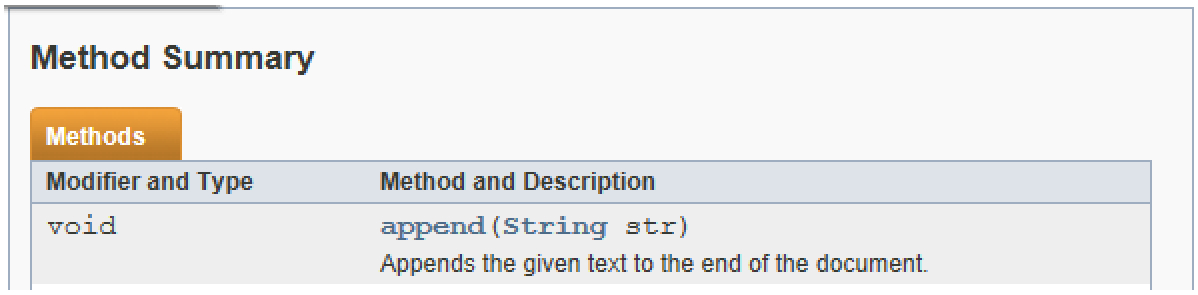
Class JTextArea



Constructor Summary

Constructors

Constructor and Description
JTextArea() Constructs a new TextArea.

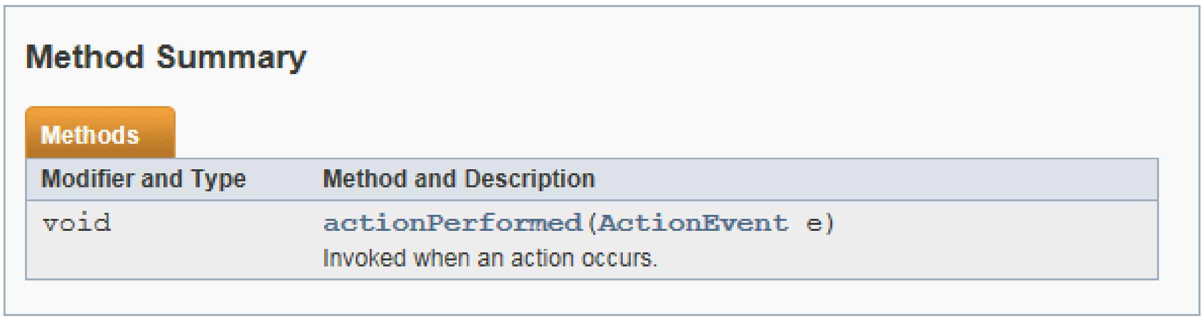


Method Summary

Methods

Modifier and Type	Method and Description
void	append(String str) Appends the given text to the end of the document.

Interface ActionListener

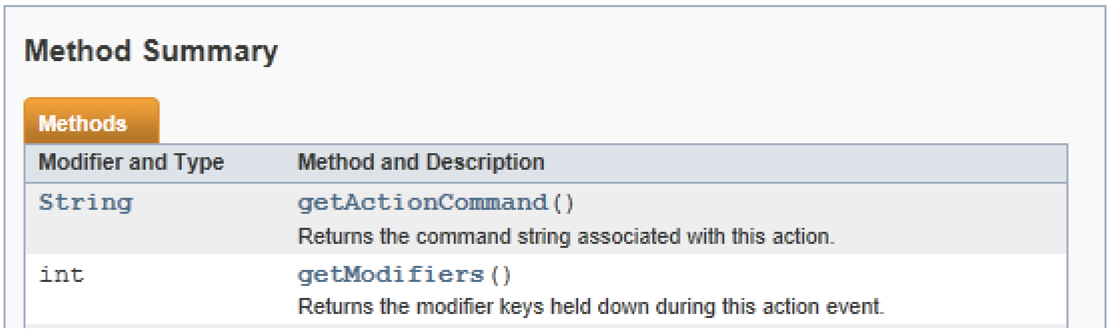


Method Summary

Methods

Modifier and Type	Method and Description
void	actionPerformed(ActionEvent e) Invoked when an action occurs.

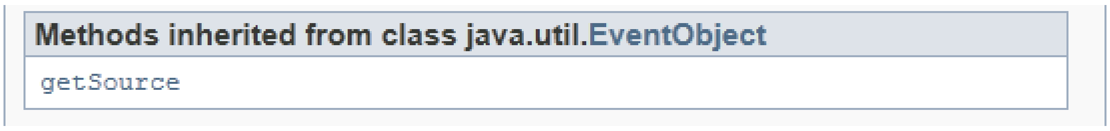
Class ActionEvent



Method Summary

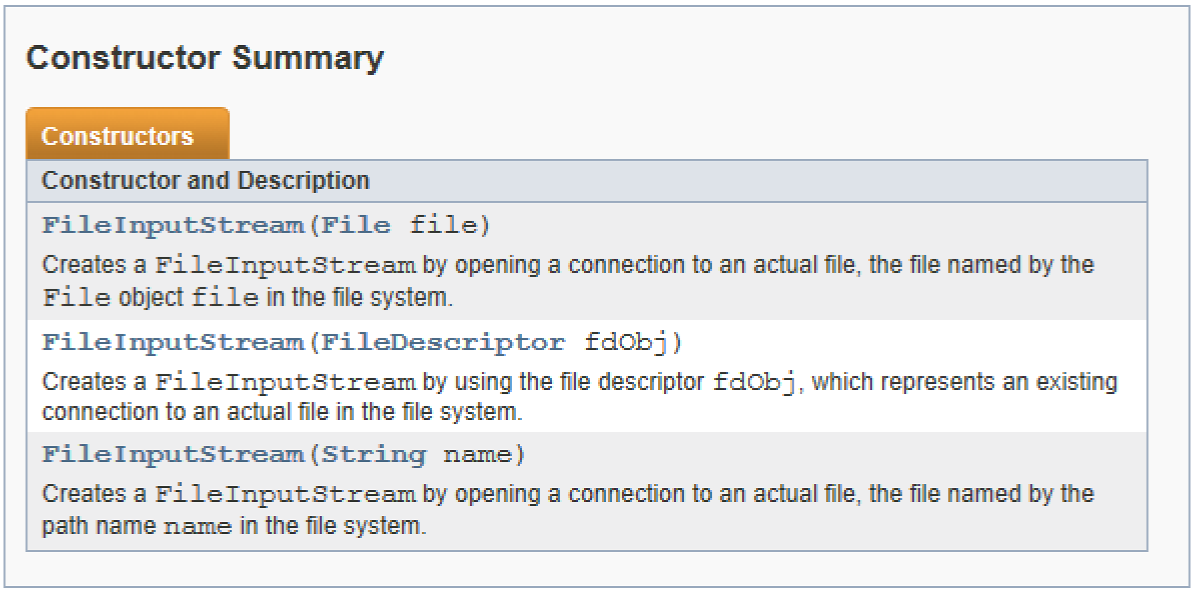
Methods

Modifier and Type	Method and Description
String	getActionCommand() Returns the command string associated with this action.
int	getModifiers() Returns the modifier keys held down during this action event.



Methods inherited from class java.util.EventObject
getSource

Class FileInputStream

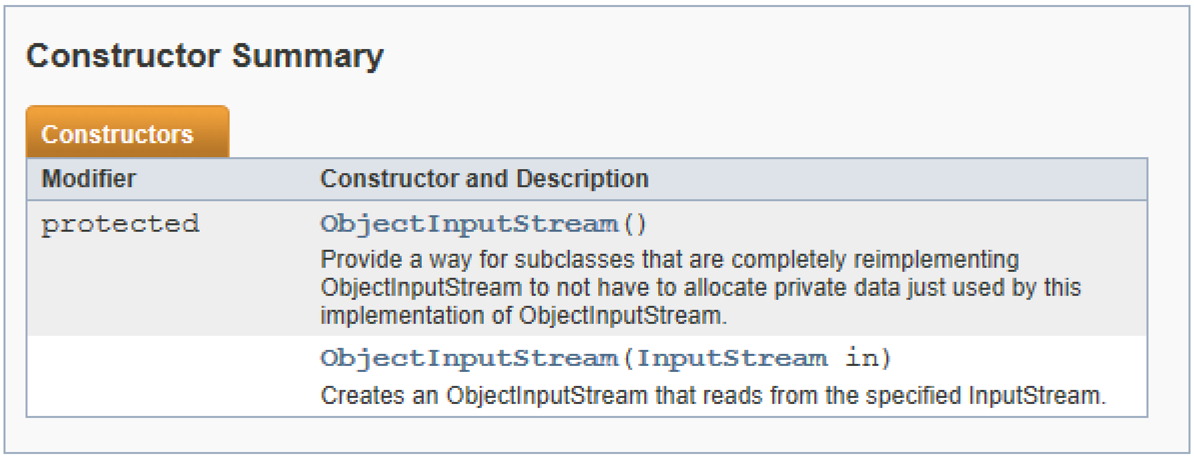


Constructor Summary

Constructors

Constructor and Description
`FileInputStream(File file)` Creates a `FileInputStream` by opening a connection to an actual file, the file named by the `File` object `file` in the file system.
`FileInputStream(FileDescriptor fdObj)` Creates a `FileInputStream` by using the file descriptor `fdObj`, which represents an existing connection to an actual file in the file system.
`FileInputStream(String name)` Creates a `FileInputStream` by opening a connection to an actual file, the file named by the path name `name` in the file system.

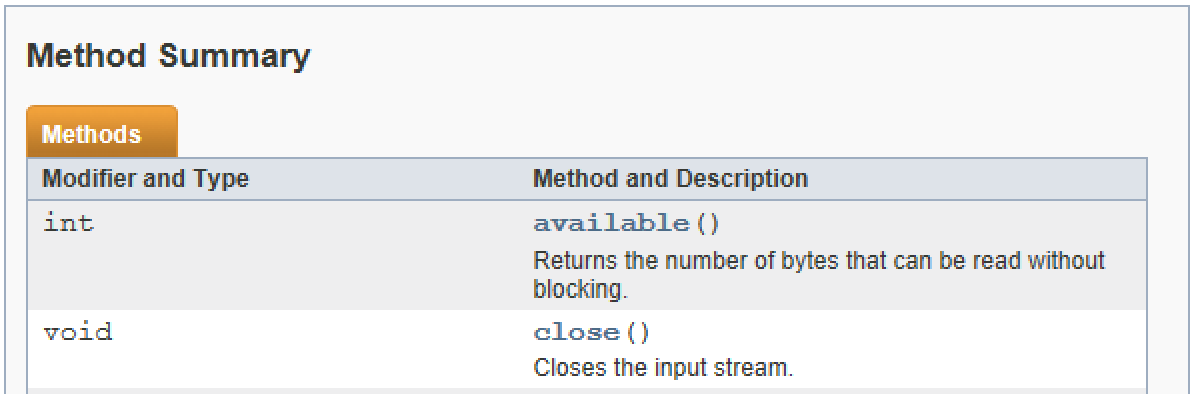
Class ObjectInputStream



Constructor Summary

Constructors

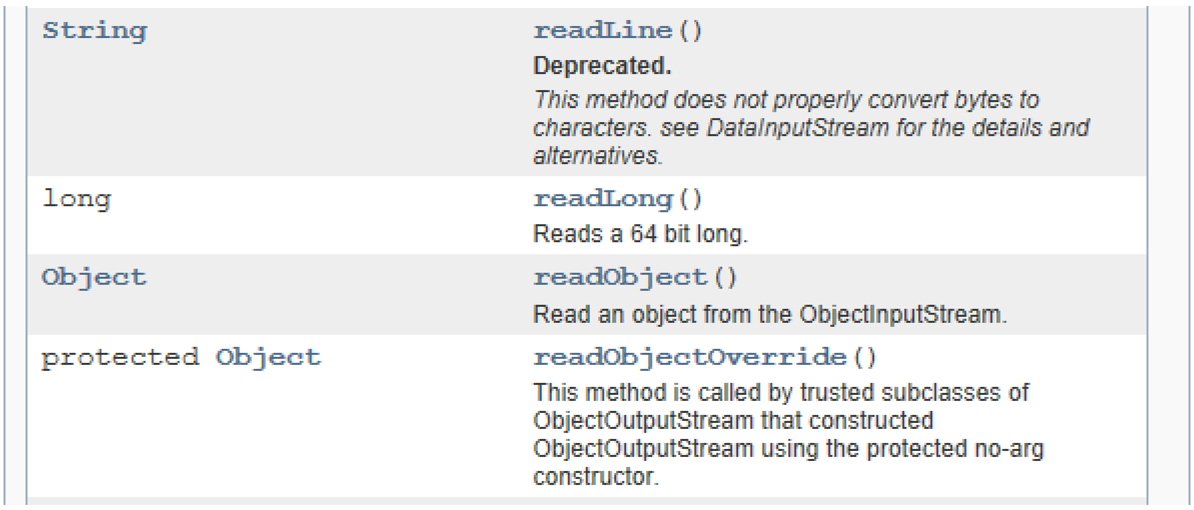
Modifier	Constructor and Description
`protected`	`ObjectInputStream()` Provide a way for subclasses that are completely reimplementing ObjectInputStream to not have to allocate private data just used by this implementation of ObjectInputStream.
	`ObjectInputStream(InputStream in)` Creates an ObjectInputStream that reads from the specified InputStream.



Method Summary

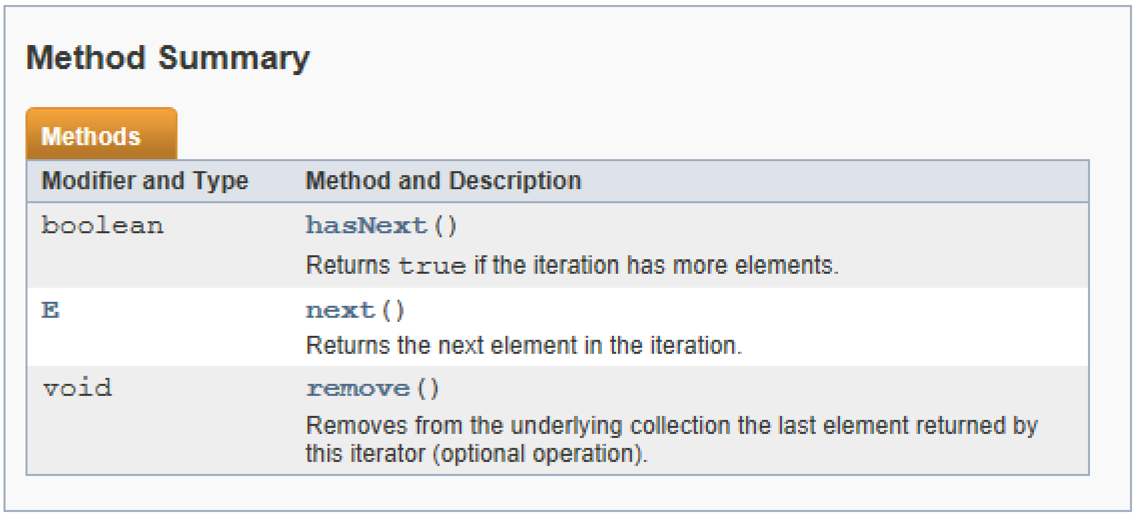
Methods

Modifier and Type	Method and Description
`int`	`available()` Returns the number of bytes that can be read without blocking.
`void`	`close()` Closes the input stream.



String	readLine() **Deprecated.** *This method does not properly convert bytes to characters. see DataInputStream for the details and alternatives.*
long	readLong() Reads a 64 bit long.
Object	readObject() Read an object from the ObjectInputStream.
protected Object	readObjectOverride() This method is called by trusted subclasses of ObjectOutputStream that constructed ObjectOutputStream using the protected no-arg constructor.

Interface Iterator<E>



Method Summary

Methods

Modifier and Type	Method and Description
boolean	hasNext() Returns `true` if the iteration has more elements.
E	next() Returns the next element in the iteration.
void	remove() Removes from the underlying collection the last element returned by this iterator (optional operation).