INSTITUTE OF TECHNOLOGY TRALEE

AUTUMN EXAMINATIONS AY 2014-2015

Object Oriented Programming 2

Module Code 61004_PROG

CRN 43841

External Examiner: Ms Sabrina Spillane

Internal Examiner: Ms John Walsh
Duration: 2 Hours

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

Use of a pencil is permitted is for writing code.

Question 1

a) Write a short note on aggregation/composition and explain how it relates to the object oriented principle of code re-use.

(4 Marks)

b) Write the class definition for a *Pet.* The *Pet* should have the following four **attributes**: type: **String**, age: **int**, vaccinations: **LinkedList** and owner:**Owner**. The class should have **mutator** and **accessor** methods defined for all attributes. The class should have a **no-argument constructor** and a **4-argument constructor**.

(15 Marks)

c) An additional **toString method** is required for the *Pet* class that outputs all the information relating to a *Pet* object is a formatted manner using the **String format** method. You can assume that the *Owner* class has an appropriate **toString** method that you can use. An example of the output is:

```
Type: Dog
Age: 3
Vaccinations: Flu, Kennel cough, Distemper,
Owner: Mary Poppins, Chimney Street
```

(8 Marks)

d) An additional class variable and method are required to track the total number of pets instantiated from the Pet class. Write the code required, indicating where any code changes should be made, to achieve this.

(6 Marks)

e) Draw a **UML** class diagram for the complete **Pet** class.

(7 Marks)

<u>Question 2</u>

The class VetSurgery.java (**Appendix 1**) is a **JFrame** based **GUI** program that allows the user to set up and maintain a **collection (LinkedList)** of pets. Each pet is an object of the Pet class developed in Question1. The menu system has **two parts**, one called ‘fileMenu’ that allows for creating, opening and saving a file as well as quitting the program, the other called ‘vetSurgeryMenu’ that allows for adding a new pet and displaying the entire **LinkedList** of pets.

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

(8 Marks)

b) Write code for the ‘addPet’ method suitable for inclusion in VetSurgery.java. The method is required to add a pet to the collection. The method should prompt the user for appropriate pet details including vaccinations via dialog boxes. If the user enters no text for a vaccination, the input of vaccinations should terminate. See **Appendix 2** for some useful information.

(12 Marks)

c) Write code for the ‘display’ method suitable for inclusion in VetSurgery.java. The method should use a **JTextArea** object to display the details of the pet and use an **iterator** to scan through the **LinkedList** to retrieve the appropriate details for each pet object. See **Appendix 2** for some useful information.

(10 Marks)

Question 3

The class VetSurgery.java (**Appendix 1**) is a **JFrame** based **GUI** program that allows the user to set up and maintain a **collection** of pets. Each pet is an object of the Pet class developed in Question1. The menu system has **two parts**, one called 'fileMenu' that allows for creating, opening and saving a file as well as quitting the program, the other called 'vetSurgeryMenu' that allows for adding a new pet and displaying the entire **LinkedList** of pets.

a) Explain how a menu system in created in Java. You should describe the step by step process indicating classes and methods used. You should also indicate how the menu is registered with an event handler object.

(8 Marks)

b) Write partial code, including the **header**, for the **actionPerformed** method that **invokes** the 'addPet' and 'display' methods when the appropriate event is detected by the program. You can assume appropriate names are given to elements of 'vetSurgeryMenu'. There is <u>no</u> requirement to write code to handle any elements of the 'FileMenu'. See **Appendix 2** for some useful information.

(8 marks)

c) Write code for the 'open' method, suitable for inclusion in VetSurgery.java, that will open a binary file called 'pets.dat' and read in an **entire LinkedList** of pets. The method should also include a try catch structure to handle any exceptions locally. See **Appendix 2** for some useful information.

(10 Marks)

d) Rewrite the 'open' method developed in part c) so that any exception is explicitly passed to the calling method.

(4 Marks)

Question 4

A standard instantiable class has been developed for a Fraction class. The class has two attributes: **num**:**int**, **denom**:**int** that represent the numerator and denominator of a fraction respectively. The class has **mutator** and **accessor** methods defined for both attributes. The class has a **2-argument constructor**, as well as a **toString** method.

a) Explain the OOP concepts of method **overloading** and method **overriding**.

(4 Marks)

b) Develop a method that adds two fractions. The statement to call the method from a driver class is **Fraction fans = f1.add(f2);** where f1 and f2 are fractions.

(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 the method developed in part b) above.

(4 Marks)

d) When the driver class is developed for the Fraction class, the user is prompted to enter a value for the numerator for a fraction using an input dialog box. Explain what would happen if the user entered a **String** instead of an **int** and write code to keep the program running until the user makes a valid entry.

(10 Marks)

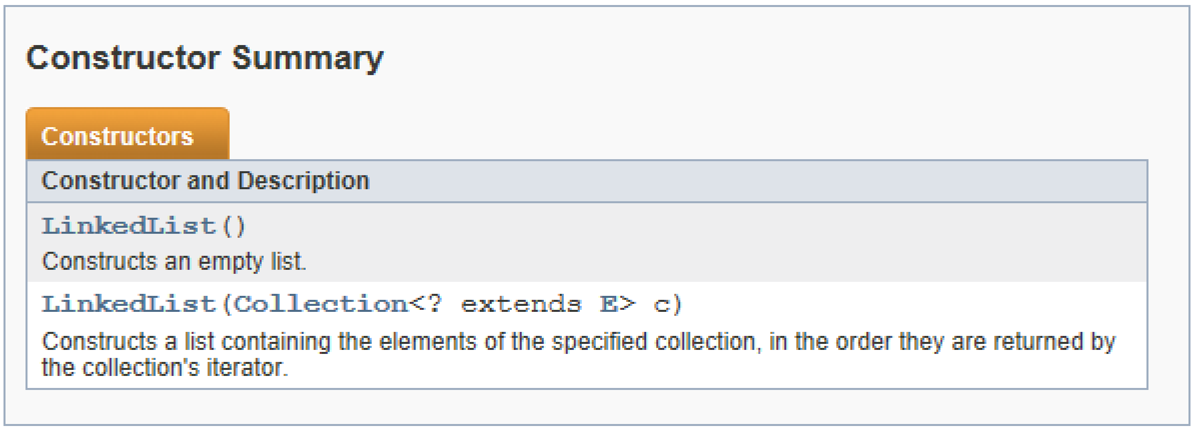
Appendix 1: VetSurgery.java

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;
import javax.swing.*;
//This program manages a collection of pets, holding the info in an LinkedList
public class VetSurgery extends JFrame implements ActionListener{
    JMenu fileMenu,vetSurgeryMenu;
    LinkedList <Pet> pets;

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

public VetSurgery( ) {
      newSystem();
      setTitle( "VetSurgery" );
      setSize( 400,200 );
      setLocation( 100,100 );
      Container pane = getContentPane();
      setDefaultCloseOperation( EXIT_ON_CLOSE );
      createFileMenu();
      createVetSurgeryMenu();
      JMenuBar menuBar = new JMenuBar();
      setJMenuBar(menuBar);
      menuBar.add(fileMenu);
      menuBar.add(vetSurgeryMenu);
   } // end

  public void newSystem() {
            pets = new LinkedList<Pet>();
   } // end new system

   public void save() throws IOException {
            //code removed
   } //end save

   public void open() {
            //code required for this method

    } // end open()

   public void addPet(){
             ///Code required for this method
    } // end addPet

     // Version of display which uses an iterator
     public void display (){
            //// code required for this method
      }
     } // end display

      private void createFileMenu(){
       // create the menu
```

```
          //code removed     }

     private void createVetSurgeryMenu(){
          // create the menu
          //code removed     }
      /** utility methods to make the code simpler */
     public void showMessage (String s){
          JOptionPane.showMessageDialog(null,s);
     }

     public void showMessage (JTextArea s){
          JOptionPane.showMessageDialog(null,s);
     }

*** actionPerformed method***

//  code required here
}
```

Appendix 2: Extracts from the Java API

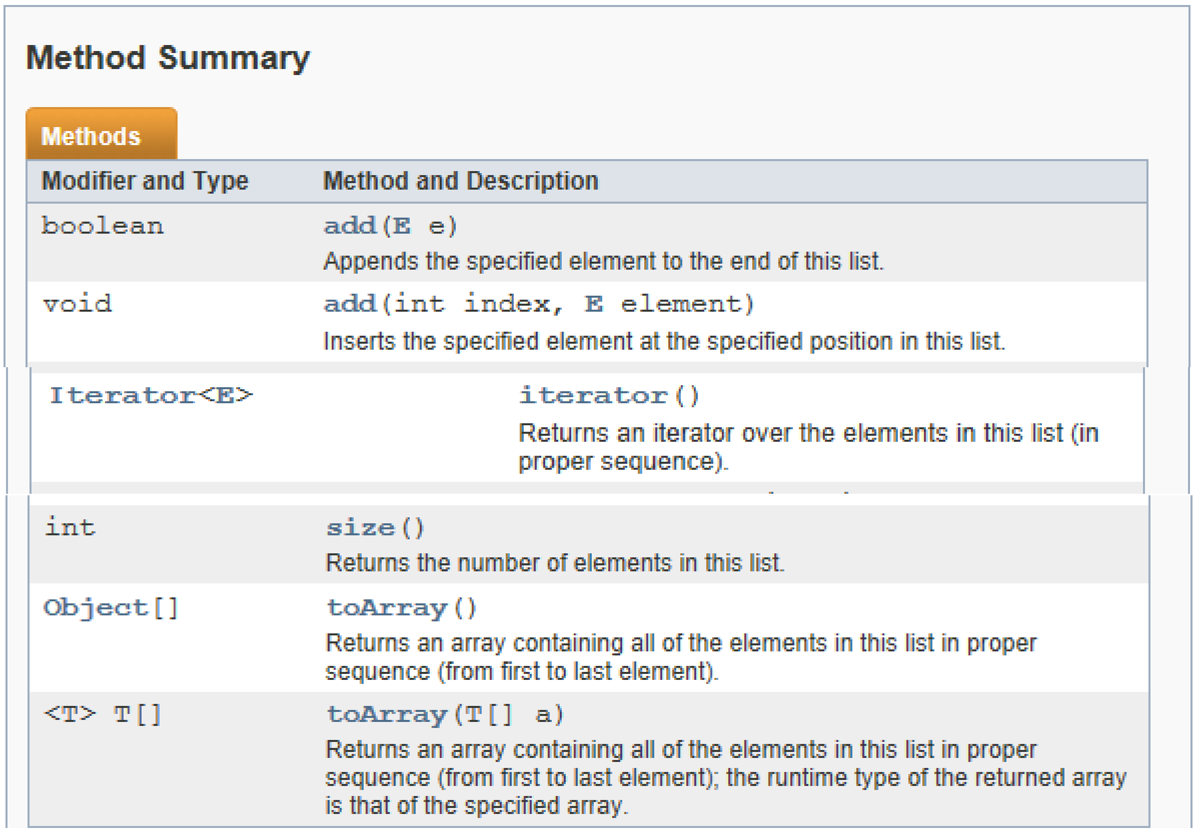
Class LinkedList<E>



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.

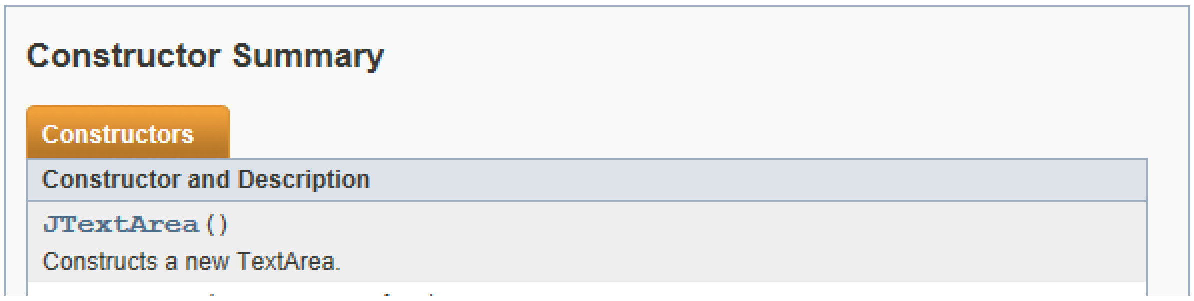


Method Summary

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.

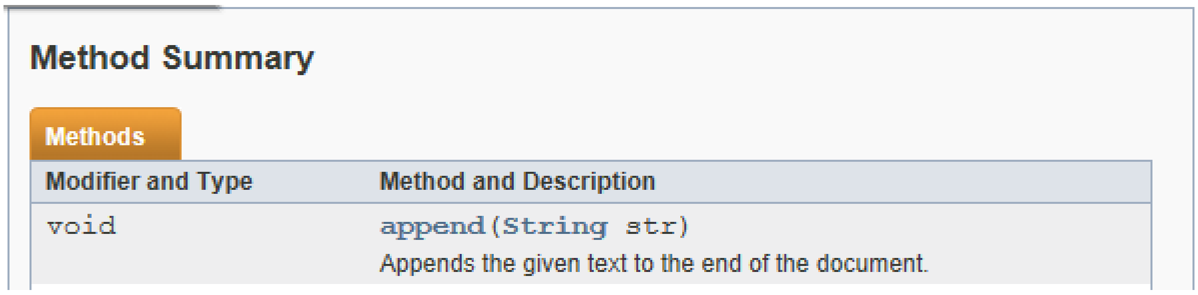
Class JTextArea



Constructor Summary

Constructors

Constructor and Description
`JTextArea()` Constructs a new TextArea.

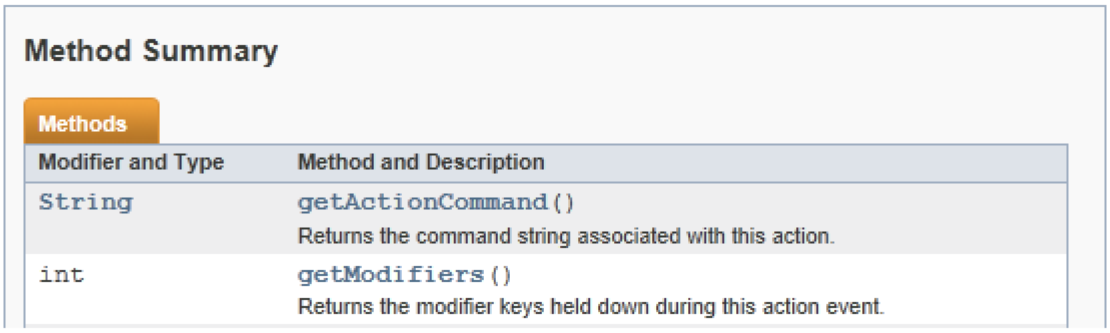


Method Summary

Methods

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

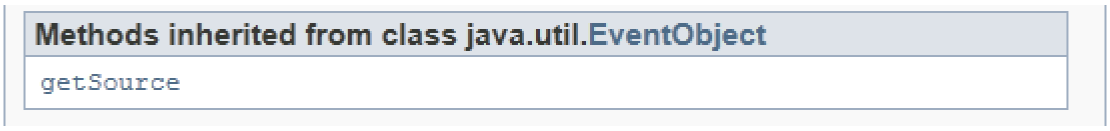
Class ActionEvent



Method Summary

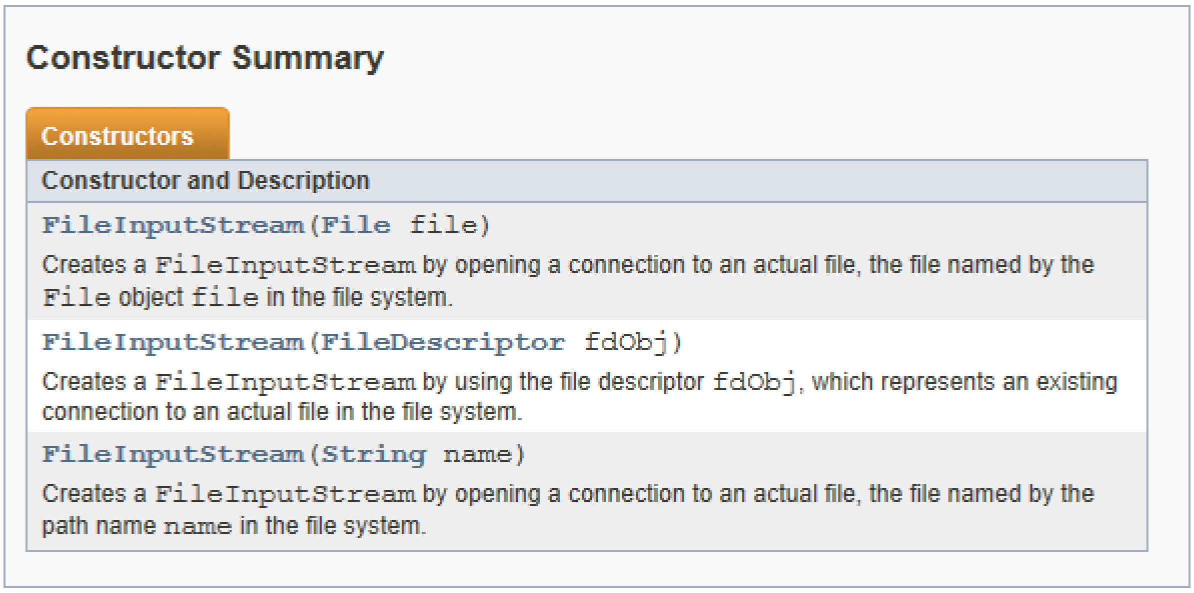
Methods

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

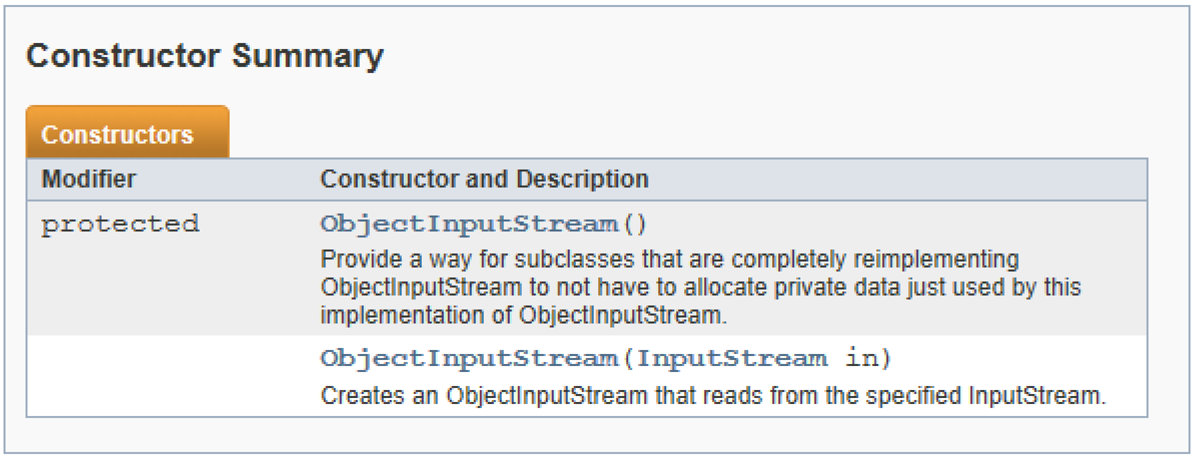


Constructor Summary

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.

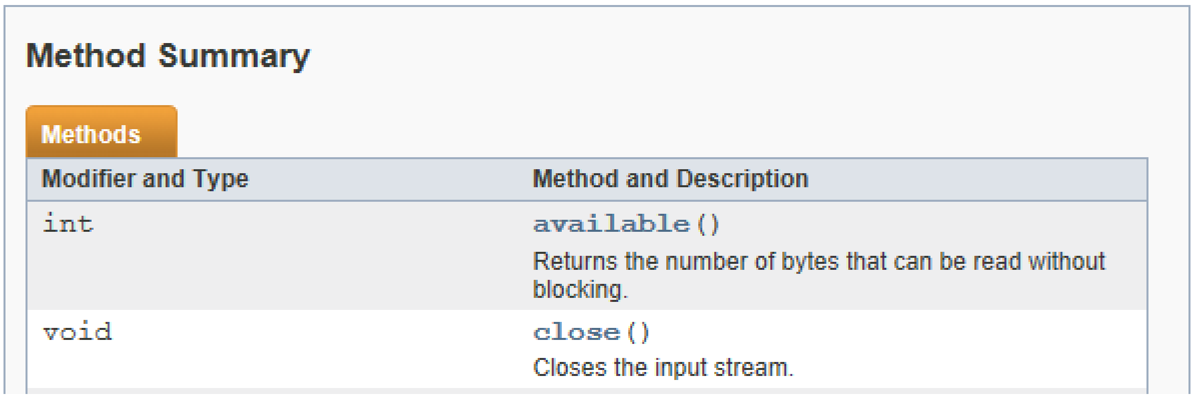
Class ObjectInputStream



Constructor Summary

Constructors

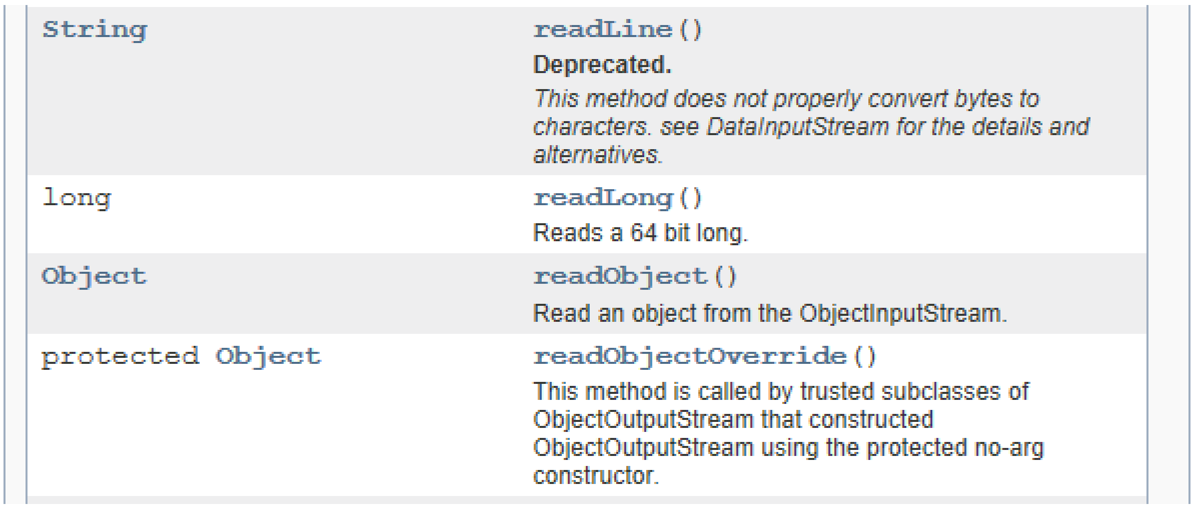
Modifier	Constructor and Description
`protected`	`ObjectInputStream()` Provide a way for subclasses that are completely reimplementing ObjectInputStream to not have to allocate private data just used by this implementation of ObjectInputStream.
	`ObjectInputStream(InputStream in)` Creates an ObjectInputStream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



Method Summary

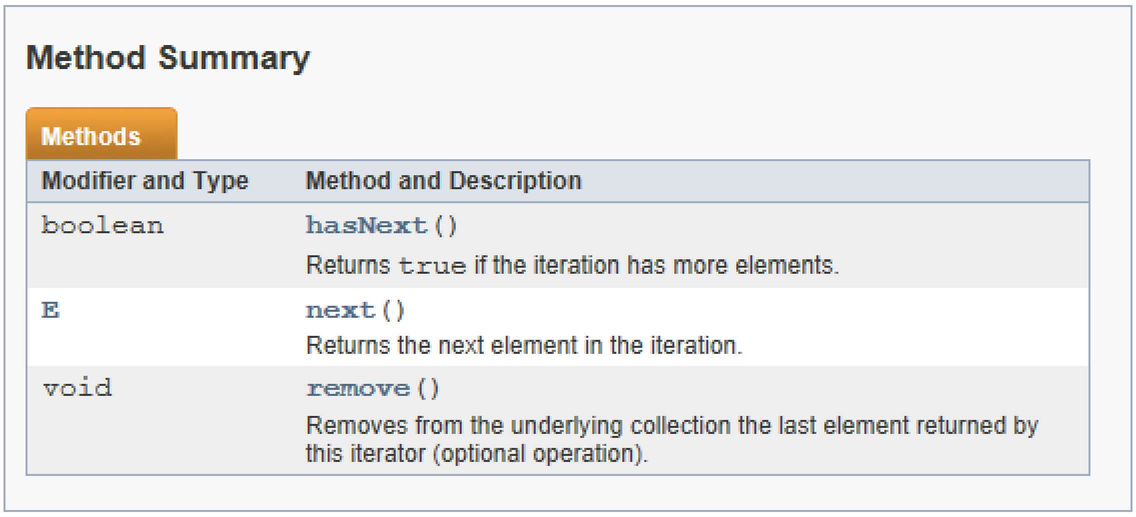
Methods

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.



Modifier and Type	Method and Description
`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).