Chapter 12

문제 12-1의 해답

○… (1) Proxy 클래스(리스트 12-6)와 Servant 클래스(리스트 12-15)는 ActiveObject 인터페이스를 구현하고 있다.

×… (2) MakerClientThread 클래스(리스트12-2)의 쓰레드가 실행하는 makeString 메소드는 Servant 클래스(리스트 12-15)에서 구현되고 있다.

➜ MakerClientThread 클래스의 쓰레드가 실행하는 makeString 메소드는 Servant 클래스가 아니라 Proxy 클래스에서 구현되고 있습니다.

×… (3) displayString 메소드를 호출할 때마다 새로운 쓰레드가 만들어진다.

×… (4) Servant 클래스(리스트 12-15)는 복수의 쓰레드에서 액세스하기 때문에 배타제어를 해야 한다.

➜ Servant 클래스에 액세스하는 것은 SchedulerThread 클래스(리스트 12-7)의 쓰레드 한 개뿐이므로 배타제어를 할 필요가 없습니다.

○… (5) ActivationQueue 클래스(리스트12-8)의 putRequest 메소드는 복수의 쓰레드로부터 호출된다.

➜ putRequest 메소드는 MakerClientThread 클래스의 쓰레드와 DisplayClientThread 클래스(리스트 12-3)의 쓰레드로부터 호출됩니다.

×… (6) ActivationQueue 클래스(리스트12-8)의 takeRequest 메소드는 복수의 쓰레드로부터 호출된다.

➜ takeRequest 메소드는 SchedulerThread 클래스(리스트 12-7)의 쓰레드 한 개에서만 호출됩니다.

○… (7) MakerClientThread 클래스(리스트12-2)의 쓰레드가 getResultValue 메소드를 호출한 시점에서 아직 문자열이 만들어지지 않았다면 쓰레드는 wait한다.

○… (8) Servant 클래스(12-15)의 makeString 메소드는 작성하는 문자열이 길수록 처리에 시간이 걸린다.

×… (9) MakerClientThread 클래스(리스트12-2)의 쓰레드가 makeString을 호출할 때 인수 count의 값이 클수록 메소드로부터 돌아오는데 시간이 걸린다.

➜ MakerClientThread 클래스의 쓰레드가 makeString을 호출할 때 실제로 문자열을 만드는 처리는 하지 않습니다. MakeStringRequest 클래스(리스트 12-10)의 인스턴스를 만들어 큐에 넣을 뿐이기 때문에 걸리는 시간은 인수 count의 값에 영향을 받지 않습니다.

문제 12-2의 해답

◆ 예제 프로그램 1의 경우

해답은 〈리스트 12-1~리스트 12-4〉와 같습니다. Chapter 12의 「메소드 추가」 순서에 따라 작업을 합니다. 참고로 java.math.BigInteger는 immutable한 클래스입니다 (Chapter 02 참조).

리스트 12-1 예제 프로그램 1을 수정한 이후의 ActiveObject 인터페이스 (ActiveObject.java)

```
package activeobject;

public interface ActiveObject {
    public abstract Result<String> makeString(int count, char fillchar);
    public abstract void displayString(String string);
    public abstract Result<String> add(String x, String y);
}
```

⇒ *예제파일 경로 : 부록CD/src/ActiveObject/A12-2a/activeobject*

리스트 12-2 예제 프로그램 1을 수정한 이후의 AddRequest 클래스(AddRequest.java)

```
package activeobject;

class AddRequest extends MethodRequest<String> {
    private final String x;
    private final String y;
    public AddRequest(Servant servant, FutureResult<String> future,
String x, String y) {
        super(servant, future);
        this.x = x;
        this.y = y;
    }
```

```
    public void execute() {
        Result<String> result = servant.add(x, y);
        future.setResult(result);
    }
}
```

⇒ 예제파일 경로 : 부록CD/src/ActiveObject/A12-2a/activeobject

리스트 12-3 예제 프로그램 1을 수정한 이후의 Proxy 클래스 (Proxy.java)

```
package activeobject;

class Proxy implements ActiveObject {
    private final SchedulerThread scheduler;
    private final Servant servant;
    public Proxy(SchedulerThread scheduler, Servant servant) {
        this.scheduler = scheduler;
        this.servant = servant;
    }
    public Result<String> makeString(int count, char fillchar) {
        FutureResult<String> future = new FutureResult<String>();
        scheduler.invoke(new MakeStringRequest(servant, future, count,
fillchar));
        return future;
    }
    public void displayString(String string) {
        scheduler.invoke(new DisplayStringRequest(servant, string));
    }
    public Result<String> add(String x, String y) {
        FutureResult<String> future = new FutureResult<String>();
        scheduler.invoke(new AddRequest(servant, future, x, y));
        return future;
    }
}
```

⇒ 예제파일 경로 : 부록CD/src/ActiveObject/A12-2a/activeobject

리스트 12-4 예제 프로그램 1을 수정한 이후의 Servant 클래스 (Servant.java)

```
package activeobject;

import java.math.BigInteger;

class Servant implements ActiveObject {
    public Result<String> makeString(int count, char fillchar) {
        char[] buffer = new char[count];
        for (int i = 0; i < count; i++) {
            buffer[i] = fillchar;
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
            }
        }
        return new RealResult<String>(new String(buffer));
    }
    public void displayString(String string) {
        try {
            System.out.println("displayString: " + string);
            Thread.sleep(10);
        } catch (InterruptedException e) {
        }
    }
    public Result<String> add(String x, String y) {
        String retvalue = null;
        try {
            BigInteger bigX = new BigInteger(x);
            BigInteger bigY = new BigInteger(y);
            BigInteger bigZ = bigX.add(bigY);
            retvalue = bigZ.toString();
        } catch (NumberFormatException e) {
            retvalue = null;
        }
        return new RealResult<String>(retvalue);
    }
}
```

⇒ *예제파일 경로 : 부록CD/src/ActiveObject/A12-2a/activeobject*

예제 프로그램 2의 경우

해답은 〈리스트 12-5~리스트 12-6〉과 같습니다.

리스트 12-5 예제 프로그램 2를 수정한 이후의 ActiveObject 인터페이스(ActiveObject.java)

```
package activeobject;

import java.util.concurrent.Future;

public interface ActiveObject {
    public abstract Future<String> makeString(int count, char fillchar);
    public abstract void displayString(String string);
    public abstract Future<String> add(String x, String y);
    public abstract void shutdown();
}
```

⇒ *예제파일 경로 : 부록CD/src/ActiveObject/A12-2/activeobject*

리스트 12-6 예제 프로그램 2를 수정한 이후의 ActiveObjectImpl 인터페이스(ActiveObjectImpl.java)

```
package activeobject;

import java.util.concurrent.Executors;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Callable;
import java.util.concurrent.Future;

import java.math.BigInteger;

// ActiveObject 인터페이스 구현 클래스
class ActiveObjectImpl implements ActiveObject {
    private final ExecutorService service =
                  Executors.newSingleThreadExecutor();

    // 서비스 종료
    public void shutdown() {
        service.shutdown();
    }

    // 반환 값이 있는 호출
    public Future<String> makeString(final int count, final char fillchar) {
        // 리퀘스트
        class MakeStringRequest implements Callable<String> {
            public String call() {
                char[] buffer = new char[count];
                for (int i = 0; i < count; i++) {
                    buffer[i] = fillchar;
```

```
                    try {
                        Thread.sleep(100);
                    } catch (InterruptedException e) {
                    }
                }
                return new String(buffer);
            }
        }
        // 리퀘스트 발행
        return service.submit(new MakeStringRequest());
    }

    // 반환 값이 없는 호출
    public void displayString(final String string) {
        // 리퀘스트
        class DisplayStringRequest implements Runnable {
            public void run() {
                try {
                    System.out.println("displayString: " + string);
                    Thread.sleep(10);
                } catch (InterruptedException e) {
                }
            }
        }
        // 리퀘스트 발행
        service.execute(new DisplayStringRequest());
    }

    // 반환 값이 있는 호출
    public Future<String> add(final String x, final String y) {
        // 리퀘스트
        class AddRequest implements Callable<String> {
            public String call() throws NumberFormatException {
                BigInteger bigX = new BigInteger(x);
                BigInteger bigY = new BigInteger(y);
                BigInteger bigZ = bigX.add(bigY);
                return bigZ.toString();
            }
        }
        // 리퀘스트 발행
        return service.submit(new AddRequest());
    }
}
```

⇒ *예제파일 경로 : 부록CD/src/ActiveObject/A12-2b/activeobject*

참고로 다음과 같이 add 메소드가 숫자 열로 인식할 수 없는 문자열을 부여했다고 생각해 봅시다.

```
Future<String> future = activeObject.add("XXX", "YYY");
```

그러면 AddRequest 클래스 안의 call 메소드가 예외 java.lang.NumberFormat Exception을 통보합니다. 그리고 그 예외는 java.util.concurrent.ExecutionException으로 랩 된 다음 Future의 get 메소드로부터 통보됩니다.

문제 12-3의 해답

2개의 해답을 소개하겠습니다.

해답 1은 java.util.concurrent 패키지를 사용하지 않고 예제 프로그램 1의 ActiveObject를 사용하여 만들었습니다. 여기에서는 검색 결과 URL을 Future 패턴(Chapter 09)으로 반환합니다. 해답 1에서는 호출하는 측(MyFrame)이 Future에 값이 설정되는 것을 기다리기 위하여 새로운 쓰레드를 사용합니다(MyFrame 클래스의 searchWord 메소드).

해답 2는 java.util.concurrent 패키지를 사용했습니다. 검색 결과 URL을 MyFrame에 통지할 때 Future 패턴을 사용하지 않고 display라고 하는 메소드로 직접 통지합니다. 해답 2에서는 Swing 프레임워크의 SwingUtilies.invokeLater가 Active Object 패턴의 Scheduler 역할 + ActivationQueue 역할을 담당하고 있는 것을 이용합니다. MyFrame 클래스는 Swing 프레임워크에 대한 Proxy 역할을 하고 있습니다.

사실 해답 2에는 「능동적인 객체」가 2개 등장합니다.

:: 능동적인 객체 (A) searcher 패키지 안에 있으면서 단어 검색이라고 하는 서비스를 제공하는 것

:: 능동적인 객체 (B) Swing 프레임워크와 MyFrame 클래스로 구성되며 MyFrame의 TextArea 클래스에 문자열을 표시하는 서비스를 제공하는 것

(A)와 (B), 이 2개의 「능동적인 객체」가 비동기 메시지에 해당하는 메소드를 서로 호출하고 있는 것입니다. 이것은 해답 2에서처럼 「능동적인 객체」가 비동기 메시지를 주고받는 모습을 표현합니다.

해답 1 : 검색한 URL을 Future 패턴으로 반환한다

ActiveObjectFactory, SchedulerThread, ActivationQueue, MethodRequestResult, FutureResult, RealResult의 각 클래스는 예제 프로그램 1과 같습니다.

리스트 12-7 해답 1 : Main 클래스 (Main.java)

```
public class Main {
    public static void main(String[] args) {
        new MyFrame();
    }
}
```

⇒ *예제파일 경로 : 부록CD/src/ActiveObject/A12-3a*

리스트 12-8 해답 1 : MyFrame 클래스 (MyFrame.java)

```
import java.io.IOException;
import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.SwingUtilities;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JButton;
import javax.swing.JTextField;
import javax.swing.JTextArea;
import javax.swing.JScrollPane;
import javax.swing.JPanel;

import activeobject.ActiveObjectFactory;
import activeobject.ActiveObject;
import activeobject.Result;

public class MyFrame extends JFrame implements ActionListener {
    private final JTextField textfield = new JTextField("word", 10);
    private final JButton button = new JButton("Search");
    private final JTextArea textarea = new JTextArea(20, 30);
    private final ActiveObject activeObject =
            ActiveObjectFactory.createActiveObject();
    private final static String NEWLINE =
            System.getProperty("line.separator");

    public MyFrame() {
```

```
        super("ActiveObject Sample");
        getContentPane().setLayout(new BorderLayout());

        // North
        JPanel north = new JPanel();
        north.add(new JLabel("Search:"));
        north.add(textfield);
        north.add(button);
        button.addActionListener(this);

        // Center
        JScrollPane center = new JScrollPane(textarea);

        // Layout
        getContentPane().add(north, BorderLayout.NORTH);
        getContentPane().add(center, BorderLayout.CENTER);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        pack();
        setVisible(true);
    }

    // Search 버튼을 눌렀을 때
    public void actionPerformed(ActionEvent e) {
        searchWord(textfield.getText());
    }

    // 표시
    private void println(String line) {
        textarea.append(line + NEWLINE);
    }

    // 검색
    private void searchWord(final String word) {
        // 검색의 호출
        final Result<String> result = activeObject.search(word);
        println("Searching " + word + "...");
        // 검색 결과를 기다리는 쓰레드
        new Thread() {
            public void run() {
                // 결과를 기다린다
                final String url = result.getResultValue();
                // 결과를 구했으므로 이벤트 디스패칭 쓰레드에 표시를 의뢰
                SwingUtilities.invokeLater(
                    new Runnable() {
```

```
                        public void run() {
                            MyFrame.this.println("word = " + word + ",
                                URL = " + url);
                        }
                    }
                );
            }
        }.start();
    }
}
```

⇒ *예제파일 경로 : 부록CD/src/ActiveObject/A12-3a*

리스트 12-9 해답 1: ActiveObject 인터페이스 (ActiveObject.java)

```
package activeobject;

public interface ActiveObject {
    public abstract Result<String> search(String word);
}
```

⇒ *예제파일 경로 : 부록CD/src/ActiveObject/A12-3a/activeobject*

리스트 12-10 해답 1: Proxy 클래스 (Proxy.java)

```
package activeobject;

class Proxy implements ActiveObject {
    private final SchedulerThread scheduler;
    private final Servant servant;
    public Proxy(SchedulerThread scheduler, Servant servant) {
        this.scheduler = scheduler;
        this.servant = servant;
    }
    public Result<String> search(String word) {
        FutureResult<String> future = new FutureResult<String>();
        scheduler.invoke(new SearchRequest(servant, future, word));
        return future;
    }
}
```

⇒ *예제파일 경로 : 부록CD/src/ActiveObject/A12-3a/activeobject*

리스트 12-11 해답 1 : SearchRequest 클래스 (SearchRequest.java)

```
package activeobject;

class SearchRequest extends MethodRequest<String> {
    private final String word;
    public SearchRequest(Servant servant,FutureResult<String> future,String word) {
        super(servant, future);
        this.word = word;
    }
    public void execute() {
        Result<String> result = servant.search(word);
        future.setResult(result);
    }
}
```

⇒ *예제파일 경로 : 부록CD/src/ActiveObject/A12-3a/activeobject*

리스트 12-12 해답 1 : Servant 클래스 (Servant.java)

```
package activeobject;

public class Servant implements ActiveObject {
    public Result<String> search(String word) {
        System.out.print("search(" + word + ")");
        for (int i = 0; i < 50; i++) {
            System.out.print(".");
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
            }
        }
        System.out.println("found.");
        String url = "http://somewhere/" + word + ".html"; // dummy URL
        return new RealResult<String>(url);
    }
}
```

⇒ *예제파일 경로 : 부록CD/src/ActiveObject/A12-3a/activeobject*

◆ 해답 2 : 검색한 URL을 MyFrame에 대한 호출로서 반환한다

리스트 12-13 해답 2 : Main 클래스 (Main.java)

```
public class Main {
    public static void main(String[] args) {
        new MyFrame();
    }
}
```

⇒ *예제파일 경로 : 부록CD/src/ActiveObject/A12-3b*

리스트 12-14 해답 2 : MyFrame 클래스 (MyFrame.java)

```
import java.io.IOException;
import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.SwingUtilities;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JButton;
import javax.swing.JTextField;
import javax.swing.JTextArea;
import javax.swing.JScrollPane;
import javax.swing.JPanel;

import searcher.Display;
import searcher.Searcher;
import searcher.SearcherFactory;

public class MyFrame extends JFrame implements Display, ActionListener {
    private final JTextField textfield = new JTextField("word", 10);
    private final JButton button = new JButton("Search");
    private final JTextArea textarea = new JTextArea(20, 30);
    private final Searcher searcher = SearcherFactory.createSearcher();
    private final static String NEWLINE=System.getProperty("line.separator");

    public MyFrame() {
        super("ActiveObject Sample");
        getContentPane().setLayout(new BorderLayout());

        // North
        JPanel north = new JPanel();
        north.add(new JLabel("Search:"));
        north.add(textfield);
        north.add(button);
        button.addActionListener(this);

        // Center
        JScrollPane center = new JScrollPane(textarea);

        // Layout
        getContentPane().add(north, BorderLayout.NORTH);
        getContentPane().add(center, BorderLayout.CENTER);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        pack();
        setVisible(true);
    }

    // Search 버튼을 눌렀을 때
    public void actionPerformed(ActionEvent e) {
```

```
        searchWord(textfield.getText());
    }

    // 표시
    private void println(String line) {
        textarea.append(line + NEWLINE);
    }

    // 검색
    private void searchWord(String word) {
        // 검색의 호출
        searcher.search(word, this);
        println("Searching " + word + "...");
    }

    // 표시
    public void display(final String line) {
        // 이벤트 디스패칭 쓰레드에 표시를 의뢰
        SwingUtilities.invokeLater(
            new Runnable() {
                public void run() {
                    MyFrame.this.println(line);
                }
            }
        );
    }
}
```

⇒ *예제파일 경로 : 부록CD/src/ActiveObject/A12-3b*

리스트 12-15 해답 2 : Display 인터페이스 (Display.java)

```
package searcher;

public interface Display {
    public abstract void display(String line);
}
```

⇒ *예제파일 경로 : 부록CD/src/ActiveObject/A12-3b/searcher*

리스트 12-16 해답 2 : Searcher 클래스 (Searcher.java)

```
package searcher;

public abstract class Searcher {
    public abstract void search(String word, Display display);
}
```

⇒ *예제파일 경로 : 부록CD/src/ActiveObject/A12-3b/searcher*

리스트 12-17 해답 2 : SearcherFactory 클래스 (SearcherFactory.java)

```
package searcher;

public class SearcherFactory {
    public static Searcher createSearcher() {
        return new SearcherImpl();
    }
}
```

⇒ *예제파일 경로 : 부록CD/src/ActiveObject/A12-3b/searcher*

리스트 12-18 해답 2 : SearcherImpl 클래스 (SearcherImpl.java)

```
package searcher;

import java.util.concurrent.Executors;
import java.util.concurrent.ExecutorService;

class SearcherImpl extends Searcher {
    private final ExecutorService service=Executors.newSingleThreadExecutor();

    public void shutdown() {
        service.shutdown();
    }

    public void search(final String word, final Display display) {
        class SearchRequest implements Runnable {
            public void run() {
                System.out.print("search(" + word + ")");
                for (int i = 0; i < 50; i++) {
                    System.out.print(".");
                    try {
                        Thread.sleep(100);
                    } catch (InterruptedException e) {
                    }
                }
                System.out.println("found.");
                String url="http://somewhere/"+word+".html";//dummy URL
                display.display("word = " + word + ", URL = " + url);
            }
        }
        service.execute(new SearchRequest());
    }
}
```

⇒ *예제파일 경로 : 부록CD/src/ActiveObject/A12-3b/searcher*