10장. 자바 메일

동의대학교 컴퓨터소프트웨어공학과

학습 목표

- 기존의 자바 플랫폼에 메일과 메시지 프레임워크를 제공하기 위해서 제안된 JavaMail API에 대하여 알아본다.
  - JavaMail API는 이메일 송수신 과정을 구현해 놓은 API로 사용자는 관련 프로토콜의 구현 방법을 몰라도 사용 가능하다.
  - 3부의 웹 메일 시스템에서 가장 많이 활용되는 API이다.

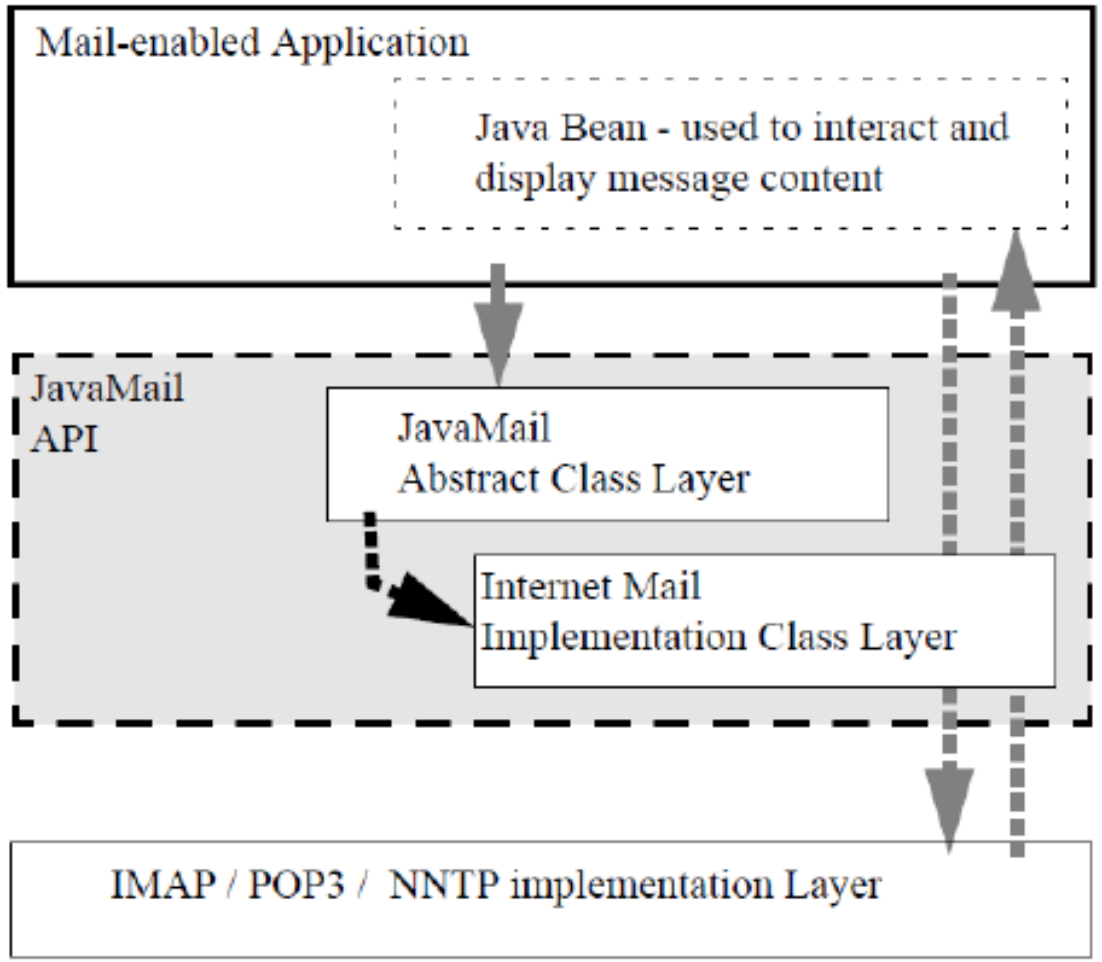
10.1 JAVAMAIL API

JavaMail API

- 참고 문헌
  - JSR-919, JavaMail API Design Specification Version 1.6.2 (2018.08.29)
  - JavaMail 1.5.0 : 2013년 4월 (17 API updates and enhancements)
  - JavaMail 1.5.6: 2016년 8월
  - JavaMail API Reference Implementation, https://javaee.github.io/javamail/
- 기존의 자바 플랫폼에 메일과 메시지 프레임워크를 제공하기 위해서 제안
- 관련 프로토콜
  - RFC 822 (SMTP), RFC 2045 (MIME), RFC 1939 (POP3), RFC 3501 (IMAP) *etc.*

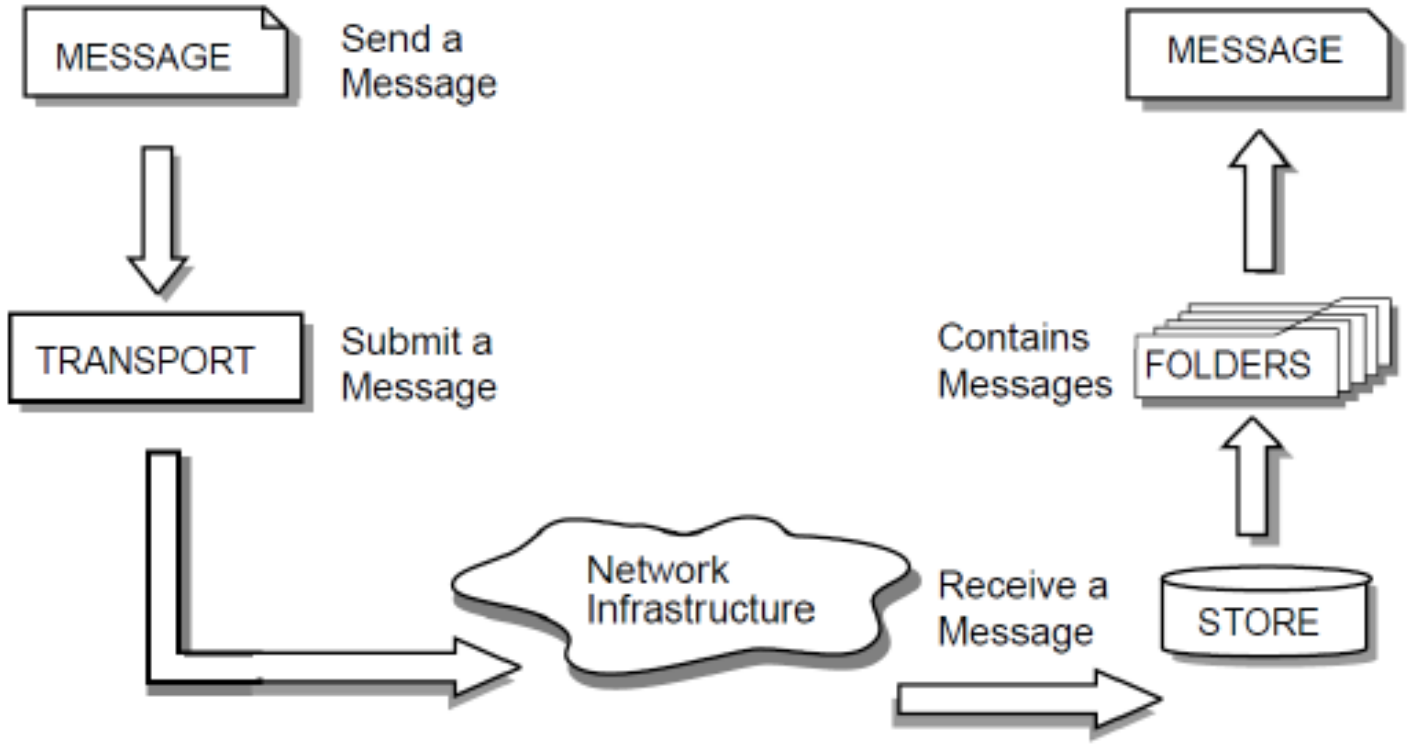
JavaMail 계층 구조

- 출처: JSR-919

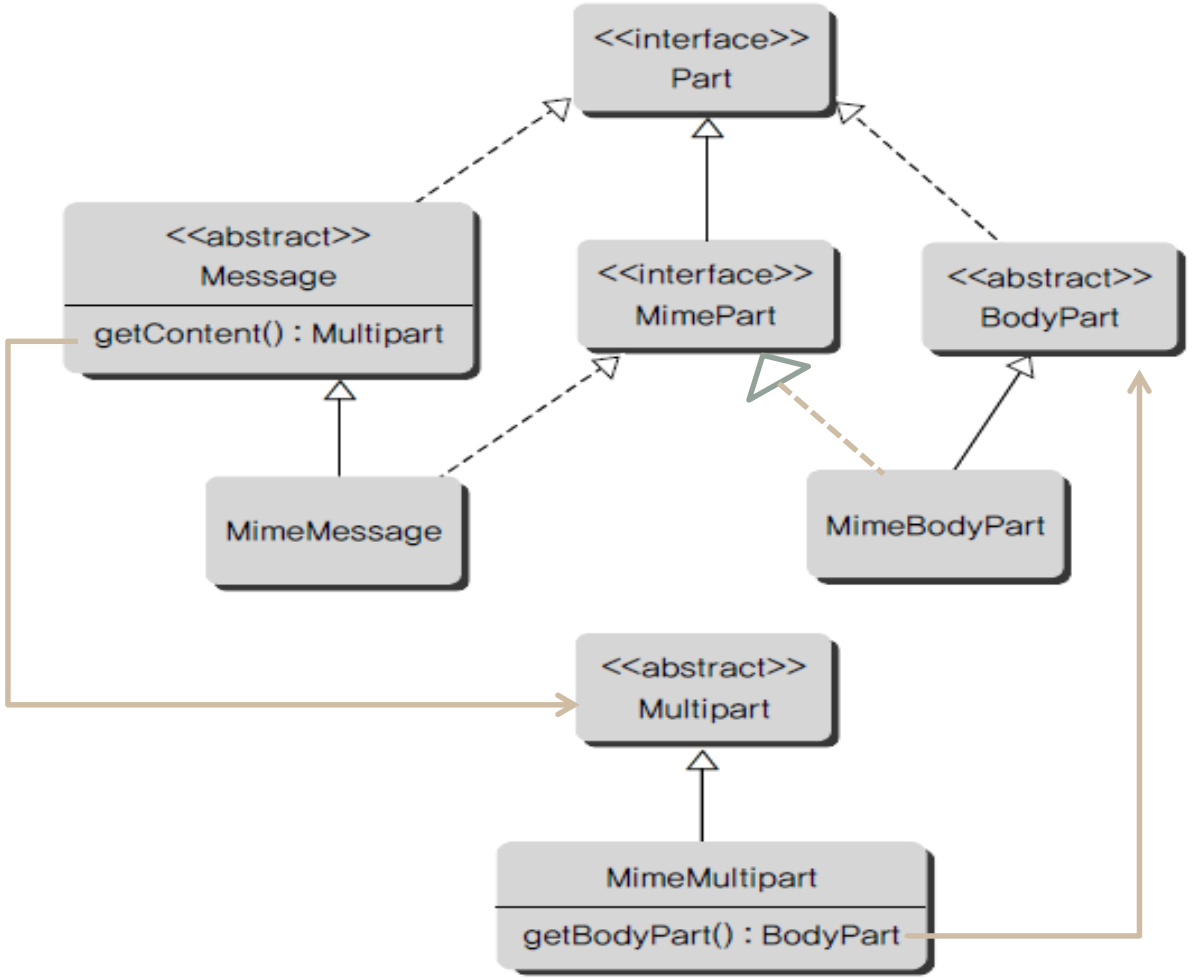


JavaMail 메시지 처리 절차

- 출처: JSR-919



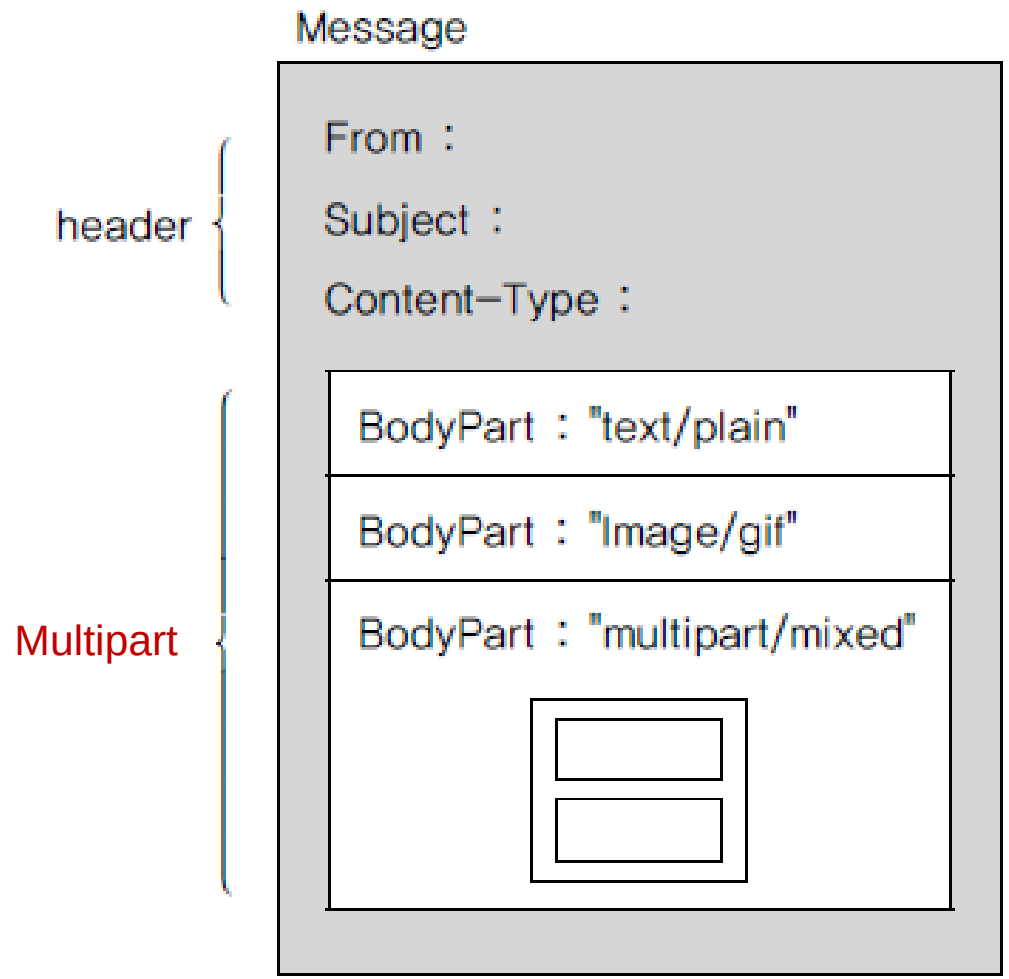
Part 인터페이스와 Multipart 클래스와의 관계



- *Part*: the common base interface for Messages and BodyParts
- *Message*: email 메시지를 모델링
- *BodyPart*: Multipart 안의 Part 모델링
- *MimePart*: RFC 2045에서 정의하는 Entity 모델링
- *MimeMessage*: MIME 형태 이메일 메시지
- *MimeBodyPart*: MimeMultipart 객체에 들어갈

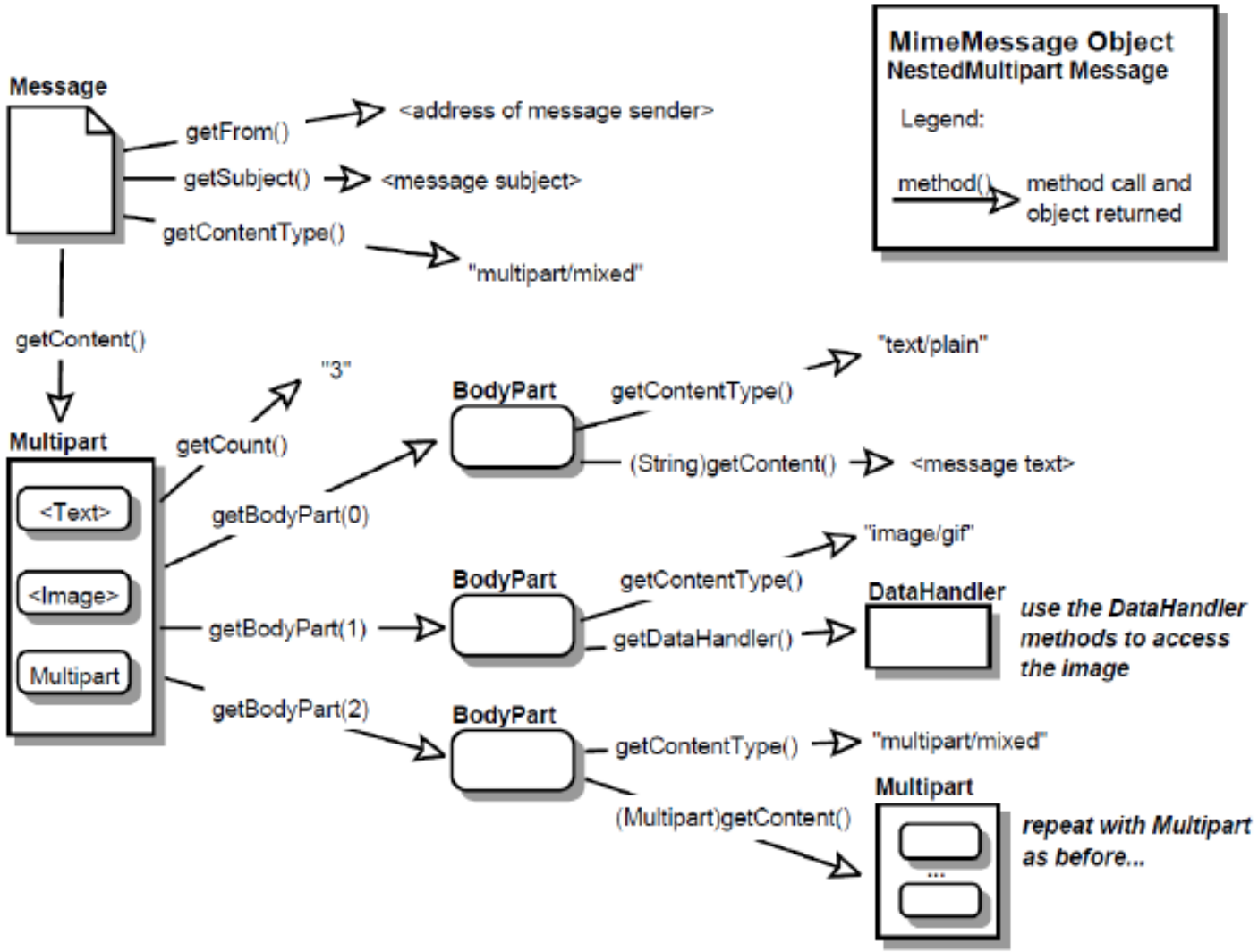
- *Multipart*: multiple body parts를 가질 수 있는 container
- *MimeMultipart*: multipart data를 위한 MIME 포맷 지원하는 Multipart 추상 클래스 구현

MimeMessage 객체의 구성 예



MimeMessage 객체 접근 방법

- 출처: JSR-919



주요 클래스

- javax.mail.Folder 추상 클래스
  - 메시지에 대한 저장 공간 관리
- javax.mail.Store 추상 클래스
  - 메시지를 저장하고 검색하는데 필요한 메시지 저장소(message store)와 접근 프로토콜을 모델링
- javax.mail.Transport 추상 클래스
  - 메시지를 목적지 주소까지 전달해 주는 전송 에이전트(transport agent)를 모델링
- javax.mail.Session 클래스
  - 메일 API에 의해서 사용되는 속성(property)과 기본 값을 관리
- javax.mail.Message 클래스
  - 송수신 메시지를 나타냄. 10.2절에서 보다 상세하게 설명.

이벤트 모델

- javax.mail.event.MailEvent 추상 클래스
  - 각 이벤트 클래스의 상위 클래스
- 이벤트 클래스
  - ConnectionEvent 클래스
  - FolderEvent 클래스
  - MessageChangedEvent 클래스
  - MessageCountEvent 클래스
  - StoreEvent 클래스
  - TransportEvent 클래스
- 각 이벤트 클래스에 대한 청취자 클래스가 있어 이벤트 발생을 감지하고 처리
- JavaBeans Spec.에 있는 JDK 1.1 이벤트-모델 명세서를 따름.
  - 출처: https://download.oracle.com/otndocs/jcp/7224-javabeans-1.01-fr-spec-oth-JSpec/

MessageCountEvent 사용 (POP3 동작?)

- javax.mail.event.MessageCountListener 인터페이스
  - Javax.mail.event.MessageCountAdapter 클래스
    - MessageCountListener 인터페이스 구현
    - This class is provided as a convenience for easily creating listeners by extending this class and overriding only the methods of interest.

Modifier and Type	Method and Description
void	**messagesAdded(MessageCountEvent e)** Invoked when messages are added into a folder.
void	**messagesRemoved(MessageCountEvent e)** Invoked when messages are removed (expunged) from a folder.

- javax.mail.Folder 클래스

void	**addMessageCountListener(MessageCountListener l)** Add a listener for MessageCount events on this Folder.

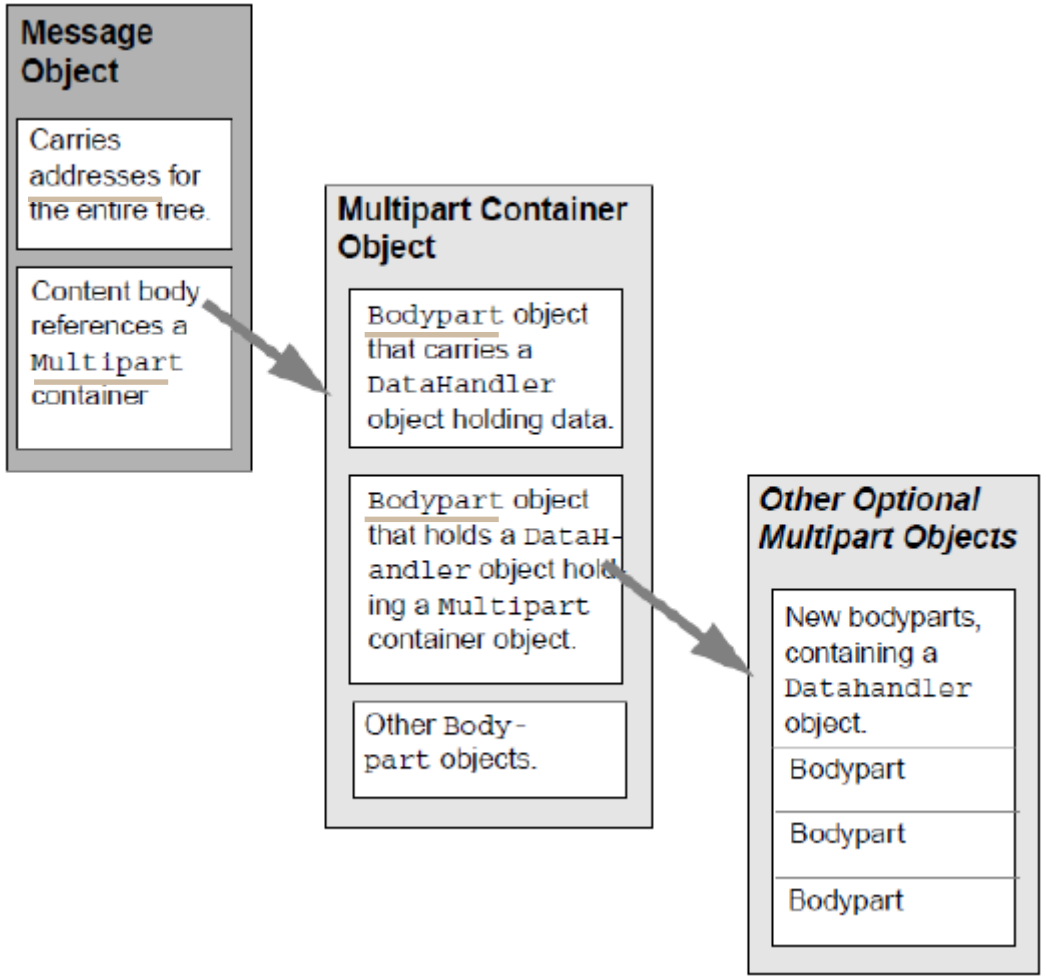
10.2 MESSAGE 클래스

Message 클래스

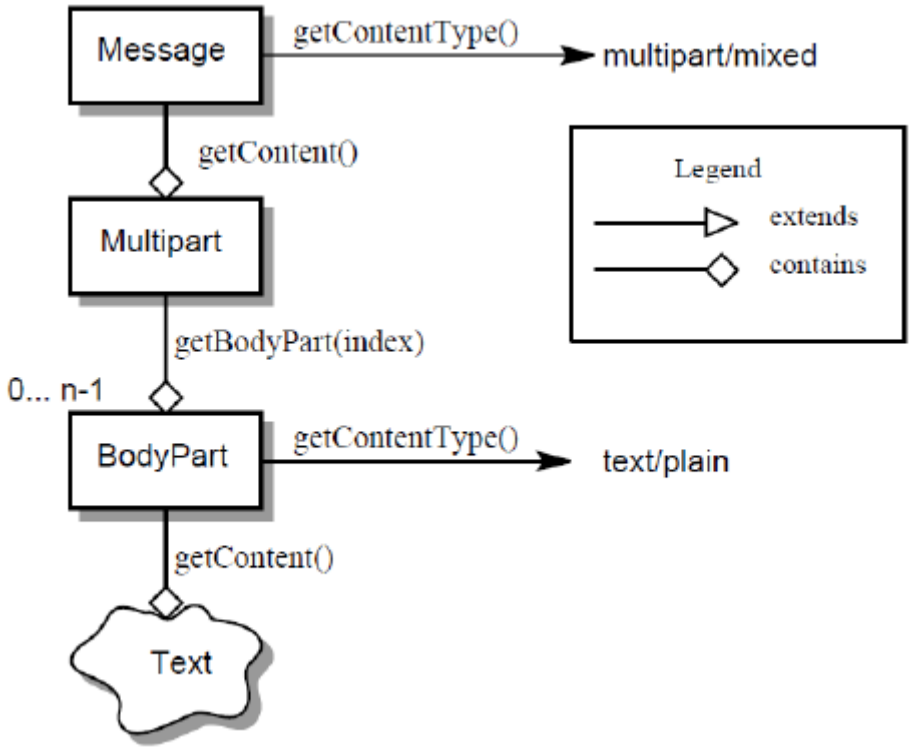
- javax.mail.Message 추상 클래스
  - 이메일 메시지를 모델링
  - javax.mail.Part 인터페이스에서 정의하는 헤더 속성 관리를 위한 addHeader, getHeader, setHeader 등의 메소드 구현
  - Message 클래스에서 새로 특정 헤더 속성을 관리하기 위한 getReplyTo, setReplyTo, getSubject, setSubject 등의 메소드를 추가

- 메시지 본문 접근
  - 메시지 본문에 접근하기 위한 메소드 없음.
    → 자신의 본문 콘텐츠에 대하여 직접적으로 접근하지 못하게 구조를 정의함으로써 임의의 콘텐츠를 본문에 포함할 수 있도록 함.
  - MimeMessage 클래스
    - Message 클래스를 상속받아 구현
    - (java.lang.Object)getContent() 메소드를 정의하여 본문 콘텐츠에 접근

Multipart Message의 구조



다편 메시지 접근 예


Message
getContentType()
multipart/mixed
getContent()
Multipart
getBodyPart(index)
0... n-1
BodyPart
getContentType()
text/plain
getContent()
Text
Legend
extends
contains

콘텐츠 본문 접근 예

다편 메시지의 콘텐츠 본문 접근

```
Multipart mp = (Multipart)o;
int count = mp.getCount();   // 몇 개의 BodyPart 객체가 있는지 알아야 함.
BodyPart bodyPart;

for (int i = 0; i < count; i++)
    bodyPart = mp.getBodyPart(i);  // 각각의 BodyPart 객체 가져와서 처리
```

10.3 메시지 작성

전송할 메시지 작성 단계

1) 메시지 생성 단계
2) 메시지 속성 설정 단계
3) 메시지 본문 설정 단계

메시지 생성

- javax.mail.Message는 추상 클래스이므로 new를 사용하여 객체 생성 불가능
  - Message의 하위 구상(concrete) 클래스인 javax.mail.internet.MimeMessage를 선택하여 사용

- 생성 예

```
Message msg = new MimeMessage(session);
```

메시지 속성 설정

- 메일 메시지의 헤더 관련 정보
  - 송신자, 수신자, 제목, 송신 날짜 등
  - 그 외 메시지 관련 헤더 정보 설정 가능
- 메시지 헤더 설정 예

메시지 헤더 설정

```
Address[] toAddrs = new InternetAddress[2];
toAddrs[0] = new InternetAddress("tester1@localhost");
toAddrs[1] = new InternetAddress("tester2@localhost");
Address fromAddr = new InternetAddress("mailtester@localhost");

msg.setFrom(fromAddr);
msg.setRecipients(Message.RecipientType.TO, toAddrs);
msg.setSubject("Test mail");
msg.setSentDate(new Date());
```

메시지 본문 설정

- 두 가지 방법
  - setDataHandler 메소드 사용
    - DataHandler 객체: 데이터를 캡슐화하는 객체
  - setContent 메소드 사용

- 사용 예

```
String content = "테스트 메일입니다.";
DataHandler data = new DataHandler(content, "text/plain");
msg.setDataHandler(data);
```

```
String content = "테스트 메일입니다.";
msg.setContent(content, "text/plain");
```

다편 메시지 생성

- 여러 BodyPart를 가지는 다편 메시지 생성 단계
  1) 새로운 MimeMultipart 객체 생성
  2) MimeBodyPart 객체를 생성하여 MimeMultipart 객체에 추가
     - 2-1) MIME 메시지의 본문을 구성하는 MimeBodyPart 객체 생성
     - 2-2) 각 BodyPart 객체의 본문을 구성하는 콘텐츠를 추가하기 위하여 setContent 메소드나 setDataHandler 메소드 사용
     - 2-3) 콘텐츠 추가가 완료되었으면 addBodyPart 메소드를 사용하여 MimeMultipart 객체에 추가
  3) setContent 메소드를 사용하여 준비가 끝난 MimeMultipart 객체를 Message 객체에 추가

다편 메시지 생성 예

MIME Multipart 메시지 만들기

```
MimeMultipart mp = new MimeMultipart();  // 1 단계

MimeBodyPart b1 = new MimeBodyPart();  // 2-1 단계
// 2-2 단계
b1.setContent("test mail: MimeBodyPart b1", "text/plain");
mp.addBodyPart(b1);  // 2-3 단계

msg.setContent(mp);  // 3 단계
msg.saveChanges();
```

* msg: 앞서 메시지 생성 단계에서 생성된 Message 객체

메시지 헤더 제약 조건

- RFC 822에 따라서 메시지 헤더는 7비트 ASCII 문자로 제한됨.
- MIME (RFC 2047)은 이러한 제약 조건을 완화시킬 수 있는 인코딩 방법 제시
- MimeUtility 클래스
  - 메시지 헤더 인코딩/디코딩 메소드 제공

javax.mail.internet.MimeUtility 클래스의 헤더 인코딩/디코딩 메소드

static String encodeText(String text);
static String encodeText(String text, String charSet, String encoding);
static decodeText(String encodedText);

메시지 본문 제약 조건

- RFC 821/822에 의하면 메시지 헤더뿐만 아니라 메시지 본문 역시 7비트 ASCII 문자로 인코딩되어야만 함.
- 해결책: MimeUtility 클래스
  - RFC 2045에서 정의하는 "base64", "quoted-printable", "7bit", "8bit", "binary" 인코딩을 지원하는 메소드를 제공

javax.mail.internet.MimeUtility 클래스의 본문 인코딩/디코딩 메소드

```
public static OutputStream encode(OutputStream os, String encoding);
public static OutputStream encode(OutputStream os, String encoding,
                                  String fimename);
public static InputStream decode(InputStream is, String encoding);
public static getEncoding(DataHandler dh);
publuc static getEncoding(DataSource ds);
```

10.4 메시지 전송

Transport 추상 클래스

- javax.mail.Transport
  - 메시지 전송을 모델링하는 추상 클래스
- com.sun.mail.smtp.SMTPTransport
  - SMTP를 사용하여 Transport 추상 클래스를 구현하는 클래스
- 메시지 전송 오퍼레이션

com.sun.mail.smtp.Transport 클래스의 메시지 전송 오퍼레이션

`static void send(Message msg);`
`static void send(Message msg, Address[] addresses);`
`abstract void sendMessage(Message msg, Address[] addresses);`

Transport 객체 접근

- javax.mail.Session 클래스 이용
  - getTransport 메소드 이용
  - cf. getStore, getProperties, *etc.*

javax.mail.Session 클래스의 Transport 객체를 가져오기 위한 메소드

`public Transport getTransport(Address address);`
`public Transport getTransport(String protocol);`
`public Transport getTransport();`

Message 객체 전송 예 1

Transport 객체를 사용한 Message 객체 전송 예

```
Session session = Session.getInstance(props, null);

Message msg = new MimeMessage(session);
//Appropriate headers should be set here.
Address[] addrs = {new InternetAddress("tester1@localhost"),
                   new InternetAddress("tester2@localhost")};
Transport trans = session.getTransport(addrs[0]);
trans.addConnectionListener(this);
trans.addTransportListener(this);

trans.connect();
trans.sendMessage(msg, addrs);
```

Message 객체 전송 예 2

- Transport.send 정적 메소드 사용

Transport 클래스의 정적 send 메소드 사용 예

```
Session session = Session.getInstance(props, null);

Message msg = new MimeMessage(session);
// Appropriate headers should be set here.
Address[] addrs = {new InternetAddress("tester1@localhost"),
                   new InternetAddress("tester2@localhost")};

Transport.send(msg, addrs);
```

SimpleSMTP 프로젝트

- 목적
  - 이메일 전송 절차 구현
- 관련 파일
  - SimpleSMTP.java
- (주의) “11.2 Apache James 메일 서버”를 참고하여 메일 서버 설치를 먼저 해야만 실행 가능. 또는 외부 네트워크에서의 SMTP 접근을 허용하는 메일 서버가 있으면 가능.

SimpleSMTP.java

- pp.276~277 [소스 10.1]
- 메일 세션 속성 설정

```
14          Properties props = System.getProperties();
15          props.setProperty("mail.debug", "true");
16          props.setProperty("mail.smtp.host", "localhost");
17          props.setProperty("mail.smtp.user", "mailtester");
```

- 메시지 객체 생성

```
19          Session session = Session.getInstance(props);
20          Message msg = new MimeMessage(session);
```

- 메시지 헤더 설정
  - 메시지 제목 설정시 7비트 ASCII 문자 제약을 해결하기 위하여 MimeUtility.encodeText 메소드 사용

```
23          Address[] toAddrs = {new InternetAddress("tester1@localhost"),
24              new InternetAddress("tester2@localhost")};
25          Address fromAddr = new InternetAddress("mailtester@localhost");
26
27          msg.setFrom(fromAddr);
28          msg.setRecipients(Message.RecipientType.TO, toAddrs);
29          msg.setSubject(MimeUtility.encodeText("메일 송신 프로그램 예제",
30                                                "UTF-8", "B"));
31          msg.setSentDate(new Date());
```

- 메시지 본문 설정

```
33          String content = "This is a test mail.";
34          msg.setContent(content, "text/plain");
```

- 메시지 전송

```
36          Transport trans = session.getTransport("smtp");
37          trans.connect();
38          trans.sendMessage(msg, toAddrs);
39          trans.close();
```

실행 결과

```
RCPT TO:<tester2@localhost>
250 2.1.5 Recipient <tester2@localhost> OK
DEBUG SMTP: Verified Addresses
DEBUG SMTP:   tester1@localhost
DEBUG SMTP:   tester2@localhost
DATA
354 Ok Send data ending with <CRLF>.<CRLF>
Message-ID: <4923951.1272549730564.JavaMail.jongmin@OfficePC1>
Date: Thu, 29 Apr 2010 23:02:10 +0900 (KST)
From: mailtester@localhost
To: tester1@localhost, tester2@localhost
Subject: =?UTF-8?B?66mU7J28IOyGoeyLoCDtlITroZzqt7jrnqgg7JiI7KCc?=
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

This is a test mail.
.
250 2.6.0 Message received
QUIT
BUILD SUCCESSFUL (total time: 0 seconds)
```

10.5 메시지 수신

Store 추상 클래스

- javax.mail.Store 추상 클래스
  - 메시지를 검색, 저장할 수 있는 접근 프로토콜과 그에 대한 메시지 저장소를 모델링
- 프로퍼티 설정
  - java.util.Properties 객체에 관련 프로퍼티 설정
  - mail.[smtp | pop3].host : 호스트 이름
  - mail.[smtp | pop3].user : 사용자 이름
  - mail.debug : JavaMail 디버깅 여부
- Store 객체 얻는 방법

javax.mail.Store 객체 얻는 방법 예

```
// props: host, user id 등의 정보를 가지고 있는 java.util.Properties 객체
POP3Store store = (POP3Store) Session.getInstance(props).getStore("pop3");
store.connect(host, user, passwd);
```

Folder 추상 클래스

- javax.mail.Folder 추상 클래스
  - 메일 메시지에 대한 폴더
  - 구현 클래스
    - com.sun.mail.imap.IMAPFolder
    - com.sun.mail.pop3.POP3Folder
- Folder 객체 얻는 메소드
  - Store 클래스에 정의

javax.mail.Store 클래스의 Folder 객체 얻기와 관련된 메소드

```
Folder getDefaultFolder();
Folder getFolder("INBOX");
Folder getFolder(URLName url);
```

- 관련 메소드

javax.mail.Folder 클래스의 Message 관련 메소드

`int getMessageCount();`
`Message getMessage(int msgno);`
`Message[] getMessages();`
`Message[] getMessages(int start, int end);`
`boolean hasNewMessages();`

메시지 헤더 출력 예

메시지 헤더 출력 예

```
POP3Store store = (POP3Store) session.getStore("pop3");
store.connect(host, user, passwd);

POP3Folder folder = (POP3Folder) store.getFolder("INBOX");
folder.open(Folder.READ_ONLY);  // or Folder.READ_WRITE

Message[] msgs = folder.getMessages();
int nth = 1;
for (Message msg : msgs) {
    MimeMessage mimeMsg = (MimeMessage) msg;
    out.append("<" + nth++ + "번째 메일 : UID = "
                        + folder.getUID(mimeMsg) + ">\n");
    out.append("From: " + mimeMsg.getFrom()[0].toString() + "\n");
    out.append("Date: " + mimeMsg.getSentDate() + "\n");
    out.append("Subject: " + mimeMsg.getSubject() + "\n\n");
}
```

메시지 헤더 미리 가져오기

- javax.mail.FetchProfile 클래스
  - 서버에 있는 메시지의 특정 속성을 미리 가져오기 위한 용도로 사용
  - add 메소드에서 미리 가져오기 위한 요소 지정
    - FetchProfile.Item.ENVELOPE
      - From, To, Cc, Bcc, ReplyTo, Subject, Date 속성
    - FetchProfile.Item.CONTENT_ INFO
      - 메시지 콘텐츠에 대한 정보 ContentType, ContentDisposition, ContentDescription, Size, LineCount 정보
    - FetchProfile.Item.FLAGS
      - 메시지 플래그 정보

- 메시지 헤더 미리 가져오기 예

메시지 헤더 미리 가져오기 예

```
Message[] msgs = folder.getMessages();

FetchProfile fp = new FetchProfile();
fp.add(FetchProfile.Item.ENVELOPE);
fp.add("X-mailer");
folder.fetch(msgs, fp);
```

SimplePOP3 프로젝트

- 목적
  - 이메일 수신 절차 구현

- 관련 파일
  - SimplePOP3.java

- (주의) “11.2 Apache James 메일 서버”를 참고하여 메일 서버 설치를 먼저 해야만 실행 가능. 또는 외부 네트워크에서의 SMTP 접근을 허용하는 메일 서버가 있으면 가능.

SimplePOP3.java

- pp.282~283 [소스 10.2]
- 속성 설정

```
19              Properties props = System.getProperties();
20              props.setProperty("mail.pop3.host", host);
21              props.setProperty("mail.pop3.user", user);
22              props.setProperty("mail.pop3.apop.enable", "true");
23              props.setProperty("mail.debug", "true");
```

- Store 객체 연결

```
25              Session session = Session.getInstance(props);
26
27              POP3Store store = (POP3Store) session.getStore("pop3");
28              store.connect(host, user, passwd);
```

- Folder 객체 읽기 모드 설정

```
33          POP3Folder folder = (POP3Folder) store.getFolder("INBOX");
34          folder.open(Folder.READ_ONLY);  // or Folder.READ_WRITE
```

- 메시지 헤더 읽기

```
37          Message[] msgs = folder.getMessages();
38          int nth = 1;
39          for (Message msg : msgs) {
40             MimeMessage mimeMsg = (MimeMessage) msg;
41             out.append("<" + nth++ + "번째 메일 : UID = "
42                      + folder.getUID(mimeMsg) + ">\n");
43             out.append("From: " + mimeMsg.getFrom()[0].toString() + "\n");
44             out.append("Date: " + mimeMsg.getSentDate() + "\n");
45             out.append("Subject: " + mimeMsg.getSubject() + "\n\n");
46          }
```

실행 결과

- pp.284~286
- Line#3~91: [소스 10.2] Line#23의 “mail.debug” 프로퍼티를 true로 설정하여 디버깅 모드로 동작하여 생성된 부분
- 메일 헤더 출력 부분

```
Date: Tue, 5 May 2020 18:40:18 +0900 (KST)
From: mailtester@localhost
To: tester1@localhost, tester2@localhost
Message-ID: <1561408618.0.1588671623614@LJM-LG-NOTEBOOK>
Subject: =?UTF-8?B?66mU7J28IOyGoeyLoCDtlITroZzqt7jrnqgg7Jil7KCc?=
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

.
QUIT
+OK Apache James POP3 Server signing off.
<1번째 메일 : UID = Mail1588672280053-2>
From: mailtester@localhost
Date: Tue May 05 18:51:11 KST 2020
Subject: 메일 송신 프로그램 예제

<2번째 메일 : UID = Mail1588672248500-0>
From: mailtester@localhost
Date: Tue May 05 18:50:40 KST 2020
Subject: 메일 송신 프로그램 예제

<3번째 메일 : UID = Mail1588671627153-0>
From: mailtester@localhost
Date: Tue May 05 18:40:18 KST 2020
Subject: 메일 송신 프로그램 예제
```

NetBeans IDE의 Run File 실행 환경 수정*

- 프로젝트 속성 > Actions > Run file via main() 수정:
  exec.args에 **-Dfile.encoding=UTF-8** 추가