Extending a CICS web application using JCICS

Extending a CICS web application using JCICS

Course introduction

What you'll see in this course

- Fundamentals of interacting with CICS
- Invoke other CICS programs
- Access CICS resources
- Units of work
- Error conditions

Sample application code

- `github.com/cicsdev`
- `cics-java-liberty-restapp-ext`
- Simple JAX-RS applications
- Only use HTTP GET verb
- Test using web browser

Assumptions

- Liberty JVM server configured
- Configured with `jaxrs-1.1` feature
- Development environment for CICS
- Dynamic web project
- Deploy using drop-ins or CICS bundles

Using Java to access CICS

Mixed-language communication

Mixed-language communication

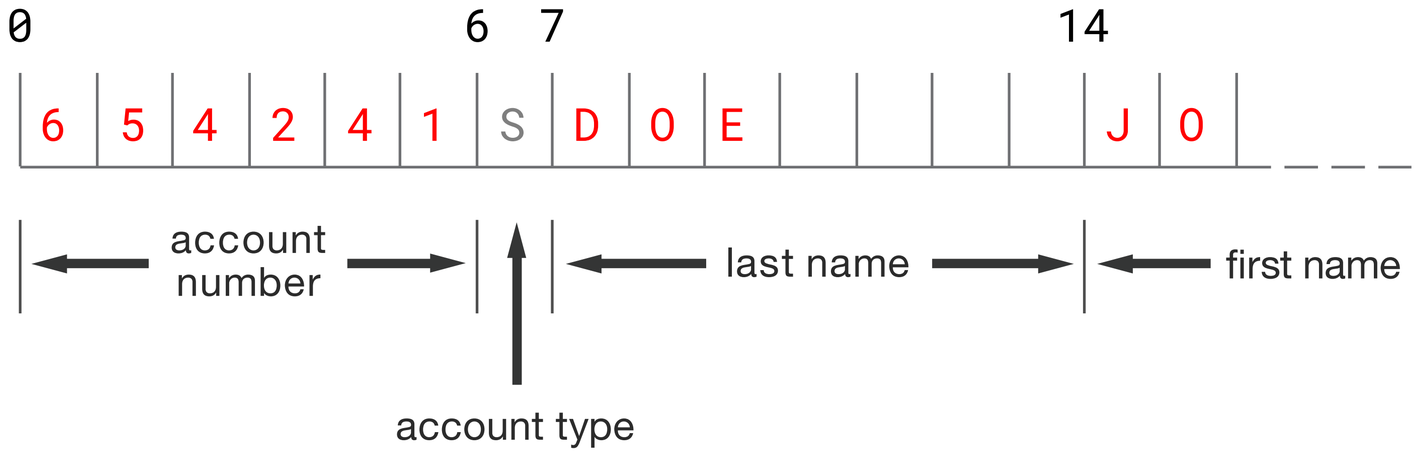
- Single CICS task sharing resources
- Program interface
  - COMMAREAs or channels
- Language-specific structures with fields
  - Define storage layout and data types

Language structures

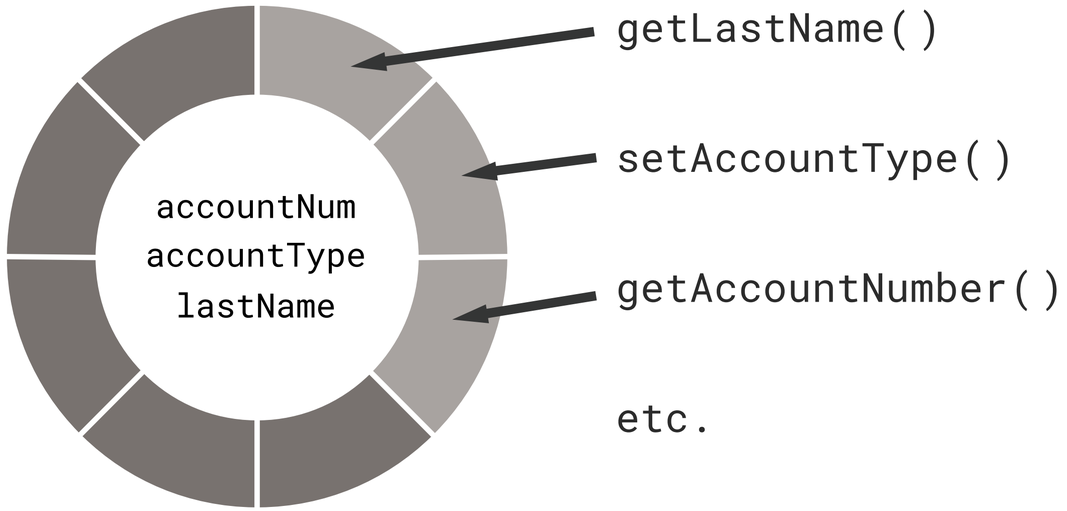
- Defined by
  - COBOL copybooks
  - C header files
- Used for sequential record data

Structured records and Java beans

Structured record



Java bean



Sample COBOL record

```
03   PART-ID                  PIC 9(8) DISPLAY.
03   SUPPLIER                 PIC 9(8) DISPLAY.
03   UNIT-PRICE               PIC 99999V99 PACKED-DECIMAL.
03   LAST-ORDER-DATE.
     05  LAST-ORDER-DATE-YY   PIC X(2).
     05  FILLER               PIC X(1) VALUE '-'.
     05  LAST-ORDER-DATE-MM   PIC X(2).
     05  FILLER               PIC X(1) VALUE '-'.
     05  LAST-ORDER-DATE-DD   PIC X(2).
03   STOCK-QUANTITY           PIC 9(8) BINARY.
03   NEXT-ORDER-DATE.
     05  NEXT-ORDER-DATE-YY   PIC X(2).
     05  FILLER               PIC X(1) VALUE '-'.
     05  NEXT-ORDER-DATE-MM   PIC X(2).
     05  FILLER               PIC X(1) VALUE '-'.
     05  NEXT-ORDER-DATE-DD   PIC X(2).
03   DESCRIPTION              PIC X(40).
```

Next steps

- JZOS record generator tool
- COBOL copybook to Java object mapping

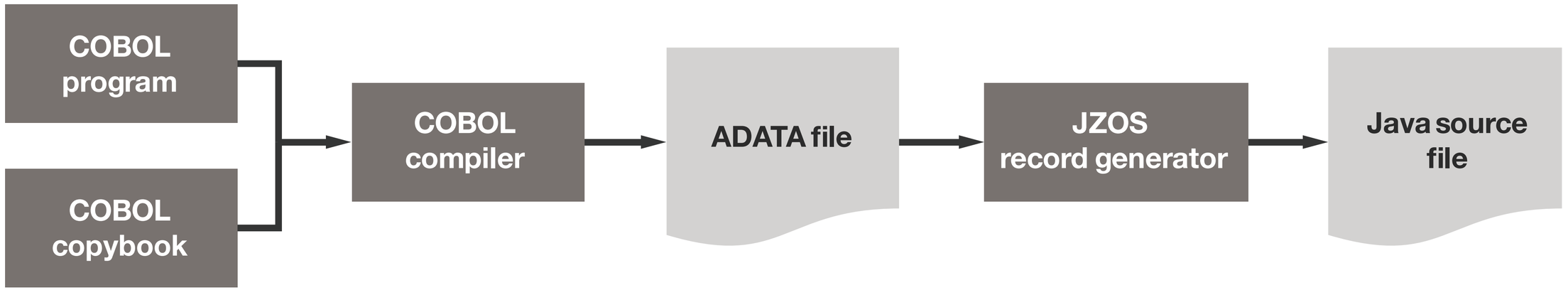
Using Java to access CICS

Java record generation

The JZOS toolkit

- Java records map native language structures
- IBM record generation tools
  - J2C record importer
  - JZOS record generator

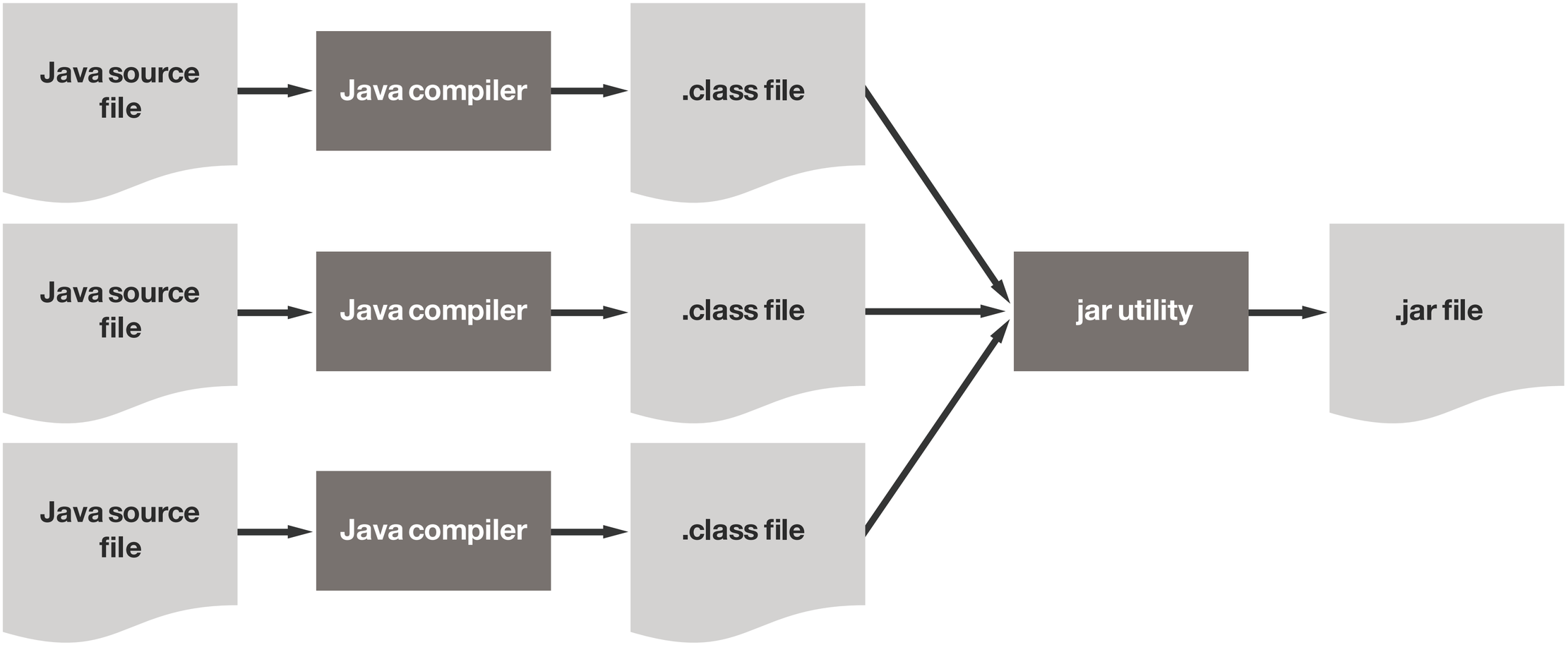
Record generation process



Including generated Java source

- Include source only when customization needed
- Import Java source file into Eclipse project
- Add JZOS library to the project build path
- Copybook change requires
  - Regeneration of source
  - Update of source in Eclipse project

Building a library JAR



Summary

- Java for CICS – Building Java records from COBOL with IBM JZOS
  - `developer.ibm.com/cics/2016/05/12/java-cics-using-ibmjzos/`

Communication with CICS

Linking to CICS programs

LINK command

- CICS program
  - Unit of compiled code
  - Implemented in any language
- LINK command
  - Call another language
  - Call another system

JCICS link example

```
Program p = new Program();
p.setName("PROG1");
p.link();
```

A COMMAREA

- Area of memory
- No structure defined to CICS
- Described using copybook or header file
- Referenced in Java as **`byte[ ]`**
- Create **`byte[ ]`** using generated Java

Example of COMMAREA and Java

```
StockPart sp = new StockPart();

sp.setPartId(12345);
sp.setSupplier(34567);
sp.setStockQuantity(100);
sp.setDescription("Small green round metal plug");

byte[] buf = sp.getByteBuffer();
```

Example of LINK using a COMMAREA

```
byte[] buf = sp.getByteBuffer();

Program p = new Program();
p.setName("PROG1");

p.link(buf);

StockPart retSP = new StockPart(buf);
```

Further use of COMMAREAs

```
byte[] buf = sp.getByteBuffer();

Program p = new Program();
p.setName("GETSUPPL");

p.link(buf);

Supplier supplier = new Supplier(buf);
```

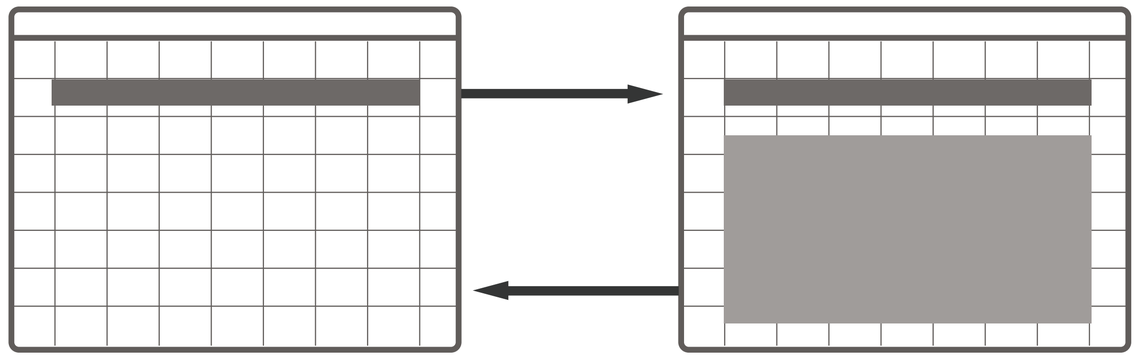
Data length optimization

```
// getByteBuffer returns length 80
byte[] buf = sp.getByteBuffer();

Program p = new Program();

p.setName("GETPART");

p.link(buf, 8);
```



Channels and containers

- Channel
  - Holds multiple containers
  - Analogous to a parameter list
- Container
  - Named block of data
  - Stores more than 32 KB of data

Summary and next steps

- Link to CICS program from Java
- Link from CICS program to Liberty

Communication with CICS

Linking to a Java program in Liberty

Linking into Java applications

- Link to Liberty
- Call Java EE applications from CICS
- POJOs invoked by CICS LINK
- Use annotation on Java method
- Auto creation of PROGRAM resource

CICSProgram annotation example

```
public class LinkToLiberty {
   @CICSProgram("GETSUPPL")
   public void getSupplierInfo() {
   …
   }
}
```

Summary

- Link to CICS programs using JCICS
- COMMAREA interface
- Link from CICS programs to Liberty

Using CICS resources

Accessing VSAM data

Introduction to CICS resources

- Temporary Storage Queue
  - Unique to CICS
- VSAM files
  - Common across z/OS

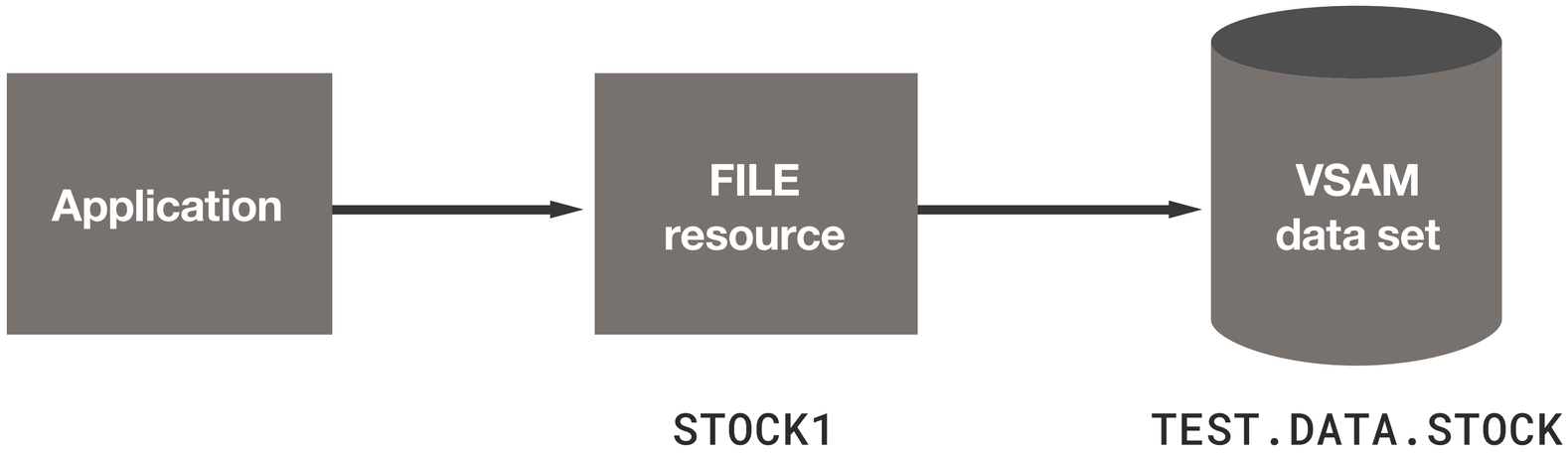
VSAM concepts

- CICS file control services
- Virtual Storage Access Method data sets
- Files shared:
  - Within a CICS region (LSR)
  - Across the sysplex (RLS)

VSAM concepts

- CICS file resource
- Unique name within CICS region
- No need to open or close file
- CICS manages file resource

Accessing a VSAM data set



VSAM data sets

- Key-sequenced data set
  - KSDS
- Entry-sequenced data set
  - ESDS
- Relative record data set
  - RRDS

IDCAMS sample input

```
DEFINE CLUSTER (                  -
    NAME ( TEST.DATA.STOCK ) -
    RECORDS ( 100 10 )            -
    INDEXED                       -
    KEYS ( 8 0 )                  -
    RECORDSIZE ( 80 80 )          -
)
```

CICS FILE resource definition

- Sample uses resource name SMPLXMPL
- Reference VSAM data set
  - DD in CICS region JCL
  - DSNAME attribute in definition
- Must enable all operations for sample
  - Add, browse, delete, read, update

RESTful interface URI

```
@ApplicationPath( "rest/" )
public class CICSApplication extends Application
{

}
```

RESTful interface URI

```
@Path( "ksds" )
@Produces( MediaType.APPLICATION_JSON )
public class VsamKsdsFileResource { }

@GET
@Path( "write" )
public StockPartCollection writeNewRecord() { }
```

/rest/ksds/write

writeNewRecord()

```
StockPart sp = StockPartHelper.generate();

byte[] record = sp.getByteBuffer();

byte[] key = StockPartHelper.getKey(sp);
```

writeNewRecord()

```
KSDS ksds = new KSDS();

ksds.setName("SMPLXMPL");

ksds.write(key, record);

Task.getTask().commit();

return queryFile(ksds);
```

updateRecord()

```
RecordHolder rh = new RecordHolder();

byte[] keyZero = StockPartHelper.getKeyZero();
ksds.readForUpdate(keyZero, SearchType.GTEQ, rh);

StockPart sp = new StockPart( rh.getValue() );
```

updateRecord()

```
StockPart spRandom = StockPartHelper.generate();
spRandom.setPartId( sp.getPartId() );

ksds.rewrite( spRandom.getByteBuffer() );

Task.getTask().commit();
```

/rest/ksds/update

deleteRecord()

```
ksds.readForUpdate(keyZero, SearchType.GTEQ, rh);

ksds.delete();
```

```
    /rest/ksds/delete
```

Summary

- Accessed VSAM KSDS files
  - Structured records
- Used generated JZOS class
  - Object-oriented model

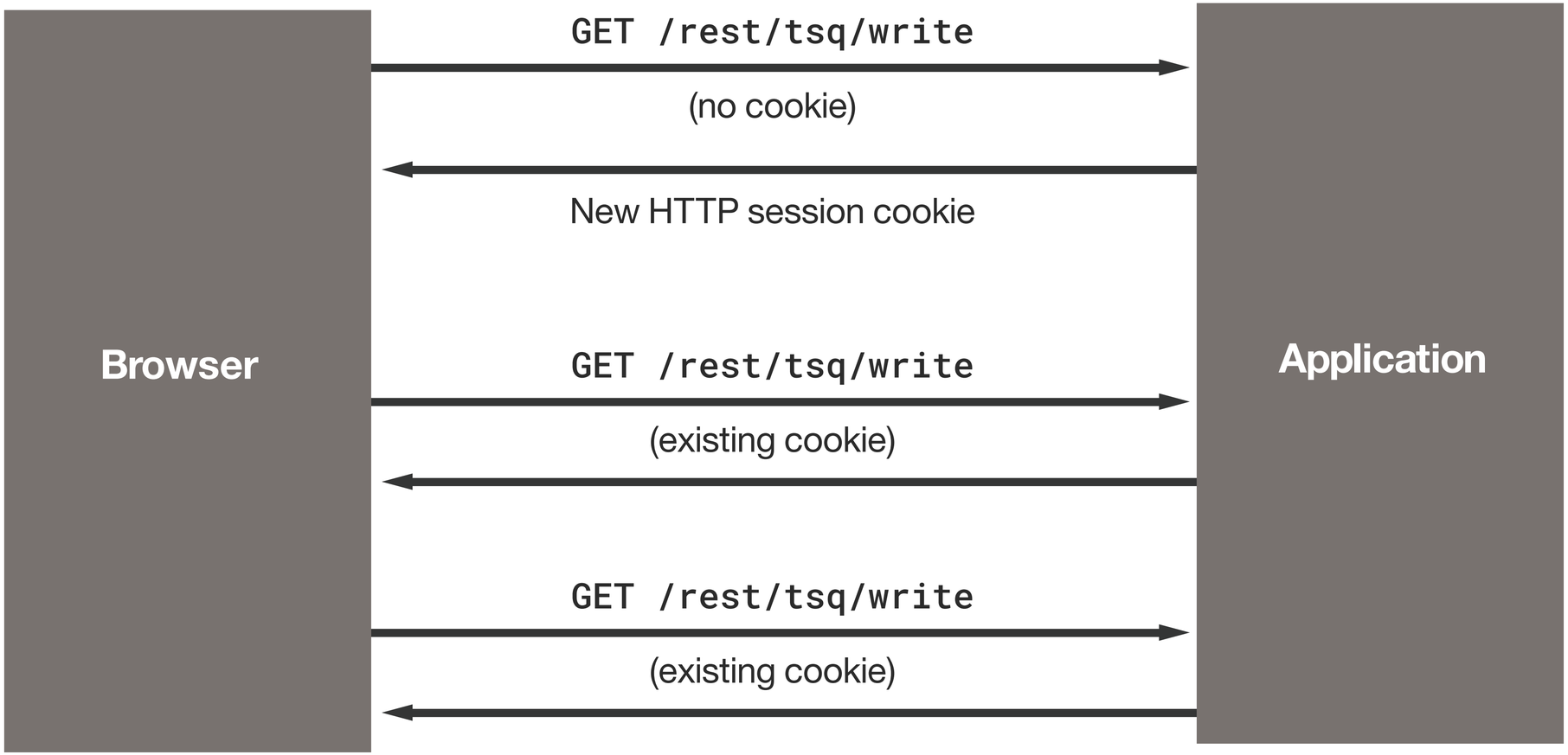
Using CICS resources

Accessing temporary storage queues

Temporary storage concepts

- Sequence of data items
- Several possible storage locations
- Each entry maximum of 32,763 bytes
- Random access
- Dynamic definition
- `com.ibm.cics.server.TSQ`

Sample TSQ application




```
public class TemporaryStorageResource
```

generateQueueName()

writeNewRecord()

```
TSQ tsq = getQueue();

StockPart sp = StockPartHelper.generate();
byte[] record = sp.getByteBuffer();

tsq.writeItem(record);

return queryQueue(tsq);
```

queryQueue()

```
ItemHolder holder = new ItemHolder();

tsq.readItem(1, holder);
```

updateRecord()

```
TSQ tsq = getQueue();
StockPart sp = StockPartHelper.generate();
byte[] record = sp.getByteBuffer();

tsq.rewriteItem(1, record);

return queryQueue(tsq);
```

deleteQueue()

```
TSQ tsq = getQueue();

tsq.delete();

return queryQueue(tsq);
```

Using CICS resources

CICS units of work

Principles of transaction processing

- Atomicity
- Consistency
- Isolation
- Durability
- Commit or rollback unit of work

CICS unit of work support

- Start of task: begin
- Normal end of task: commit
- Abnormal end of task: rollback

Using JCICS to manage the unit of work

- `Task t = Task.getTask()`
- `t.commit()`
- or
- `t.rollback()`

Summary

- Units of work
- VSAM files
- Temporary storage queues
- Transient data queues

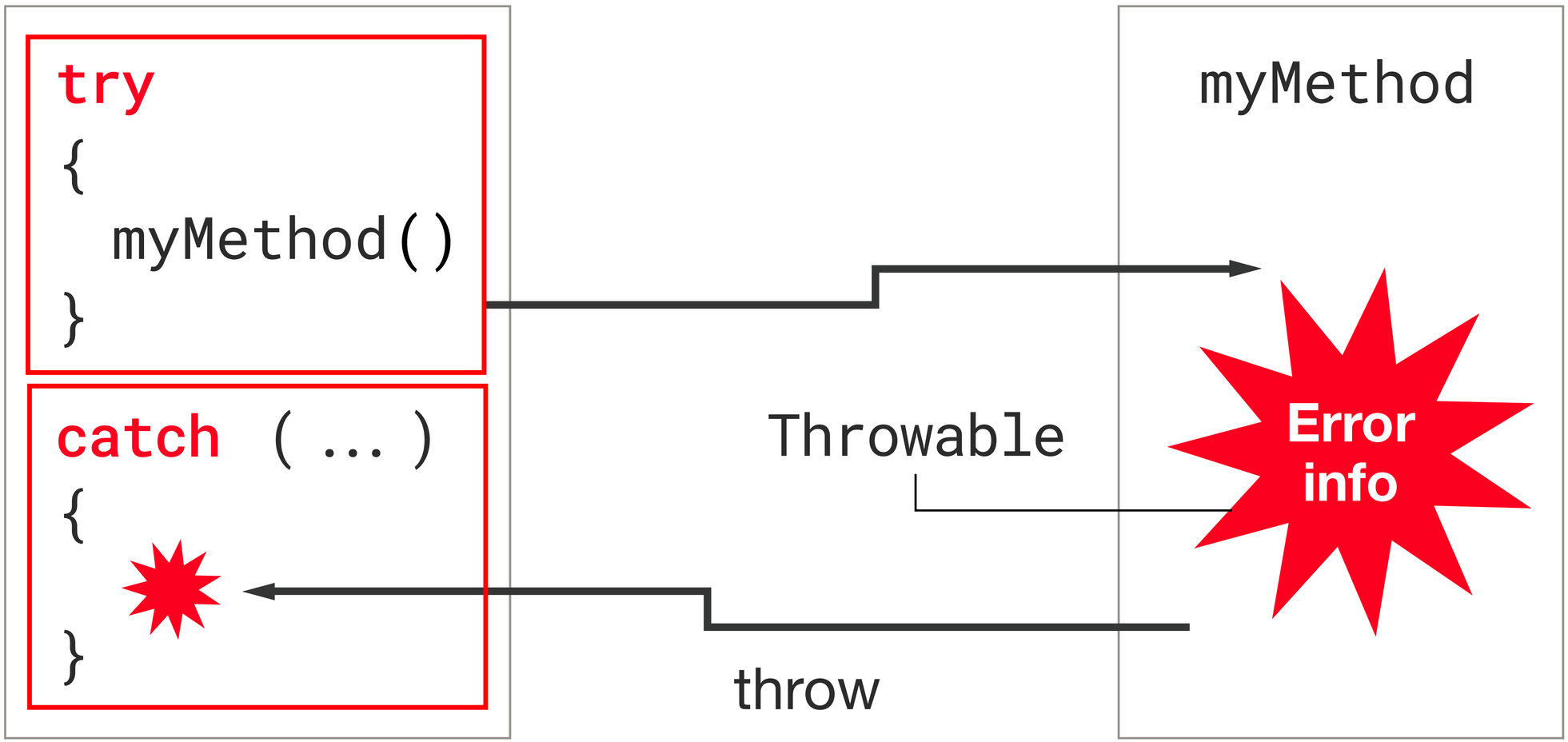
Coping with errors

Error and exception handling

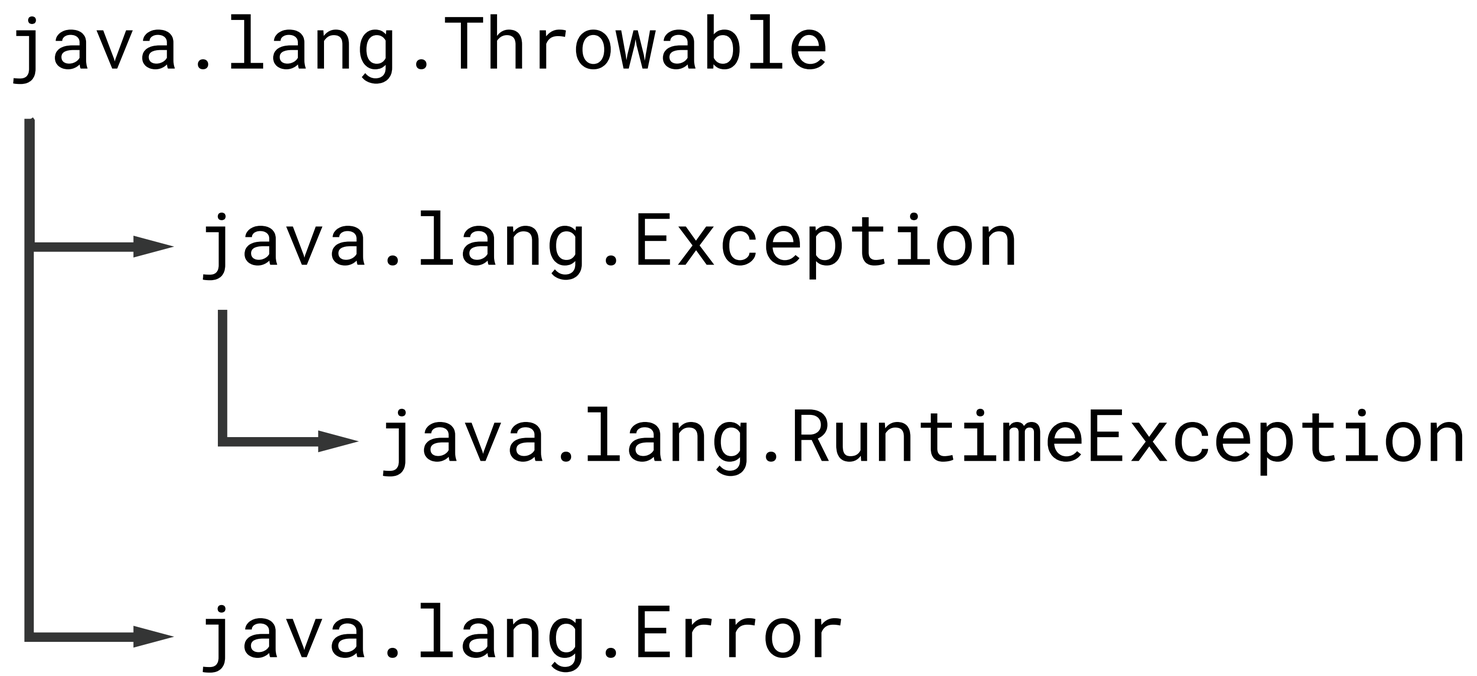
Types of errors

- Expected errors
- Unexpected errors
- Fatal errors

Throwing and catching



Java exceptions



Checked and unchecked exceptions

- `java.lang.Exception`
  - Checked exception
- `java.lang.RuntimeException`
  - Unchecked exception
- `java.lang.Error`
  - Fatal error

CICS command error handling

1. Per-command response code
   - EXEC CICS READQ TS … RESP(resp-data)
2. Active condition handler
   - EXEC CICS HANDLE CONDITION QIDERR(error-handler)
3. Active abend handler
   - EXEC CICS HANDLE ABEND LABEL(error-handler)

CICS condition exceptions

```
java.lang.Exception
    +- c.i.c.s.CicsException
       +- c.i.c.s.CicsConditionException
          +- c.i.c.s.CicsResponseConditionException
             +- c.i.c.s.ItemErrorException
             +- c.i.c.s.InvalidQueueIdException
             ...
```

c.i.c.s = com.ibm.cics.server

CICS condition exceptions

```
public int readItem(int,ItemHolder)

        throws ItemErrorException,        <-- ITEMERROR
               InvalidQueueIdException,   <-- QIDERR
               ...
```

```
TemporaryStorageResource.queryQueue(TSQ)
```

Runtime exceptions

```
java.lang.RuntimeException
+- c.i.c.s.CicsRuntimeException
   +- c.i.c.s.AbendCancelException
   +- c.i.c.s.AbendException
   +- c.i.c.s.CicsThreadingRuntimeException
   +- c.i.c.s.EndOfProgramException
   +- c.i.c.s.TransferOfControlException
```

c.i.c.s = com.ibm.cics.server

Catching exceptions

```
try { … } catch ( Exception e ) {
    // Ignore and continue
}
```

Do not use

```
try { … } catch ( Exception e ) {
    // Log and rethrow
    logger.log(e);
    throw e;
}
```

Acceptable

Fatal errors

Propagating exceptions to CICS

- `AJxx abend codes`

- EXEC CICS ABEND
  - abend(), abend(String), abend(String, boolean)
- EXEC CICS ABEND CANCEL
  - forceAbend(…)

Exception handling example

```
try {
    // Delete the record we have just read
    ksds.delete();
}
catch (RecordNotFoundException rnfe) {
    // Initial browse failed - no records in file
}
catch (CicsConditionException cce) {
    // Some other CICS failure
    throw new InternalServerErrorException(cce);
}
```

Summary

- Exceptions
  - Checked and unchecked
  - Throwing and catching
- CICS error conditions and abends
- Abends and Java exceptions

Summary

Course review

Course review

- Invoke CICS programs from Liberty
- Used JCICS to access
  - VSAM files
  - TSQs
  - Unit of work support
- Error handling