Ввод-вывод

часть 2

Сериализация

- Сериализация объекта – это сохранение объекта в поток в виде набора байтов
- Восстановление объекта из набора байтов – это десериализация
- Сериализация и десериализация должны идти по одним правилам
- Сериализация нарушает безопасность объекта
- Механизм легковесного долговременного хранения

Сериализация

- Библиотеки для сериализации (Java JDO)
- Фреймворк Hibernate
- Сериализация необходима для RMI и JavaBean

Интерфейс

- Serializable
- не содержит ни одного метода и служит флагом, помечающим класс, для которого возможна сериализация/десериализация
- когда сериализуемый объект содержит ссылки на другие объекты, будет предпринята попытка сериализовать также и их (глубокое копирование – создание дубликата графа объектов)
- Если сериализация невозможна *NotSerializableException*

Методы

- Ввод/вывод - потоки FileInputStream/FileOutputStream
- поток-обертка ObjectOutputStream
  - метод writeObject()
- поток-обертка ObjectInputStream
  - метод readObject()
- В выходной поток выводятся все нестатические поля объекта, независимо от прав доступа к ним
- для восстановления объекта необходимо иметь доступ к реализации класса этого обекта

Пример

```
class Sample implements Serializable {
    String name;
    int age;
    Sample(String name, int age){
        this.name = name;
        this.age = age;
    }
    public String getName(){
        return this.name;
    }
…
}
```

Запись

```
try{
// Создаем два сериализуемых объекта
Sample original1 = new Sample(“Bob”,”21”);
Sample original2 = new Sample(“Alice”,”20”);
FileOutputStream fos = new
        FileOutputStream(“Serialized.obj”);
ObjectOutputStream oos = new ObjectOutputStream(fos);
 // Запись двух объектов в файл
    oos.writeObject(original1);
    oos.writeObject(original2);
    oos.close();
 }catch(Exception e){ . . .}
```

Чтение

```
try{
    FileInputStream fis = new
        FileInputStream(“Serialized.obj”);
    ObjectInputStream ois = new ObjectInputStream(fis);
// Читаем записанные объекты
    Sample new1 = (Sample) ois.readObject();
    Sample new2 = (Sample) ois.readObject();
    ois.close();
    System.out.println(“Name 1 =”+new1.getName());
    System.out.println(“Name 2 =”+new2.getName());
}catch(Exception e){. . .}
```

Управление сериализацией

- В класс, реализующий интерфейс Serializabe добавить свои методы

  writeObject()

  readObject()

!!! жестко фиксированная сигнатура

- Класс, реализующий интерфейс Externalizable, реализовав свои методы сериализации

  writeExternal()

  readExternal()

Частичная сериализация

- Если необходимо предохранить часть полей объекта от записи в файл/сеть, необходимо пометить эти поля модификатором transient.

public transient String name;

Долговременное хранение

- Сохранить состояние программы для последующего восстановления
- Как решать проблему множественных ссылок на один объект?
- Когда сохранять?
- Если проводить сериализацию в единый выходной поток, сохранится вся сеть объектов и она будет гарантированно восстановлена без излишних повторений
- Безопасное сохранение – атомарной операцией. Поместить все объекты в контейнер и сохранить одной операцией.

Сжатие данных

- Пакет java.util.zip.*
- Форматы GZIP и ZIP для отдельных файлов
- Фильтры
  - GZIPOutputStream/GZIPInputStream
  - ZipOutputStream/ZipInputStream
- Многофайловые архивы (классы)
  - ZipEntry
  - ZipFile
  - CRC32
  - Adler32

Java ARchives (JAR)

- Формат ZIP
- Файл, содержащий набор сжатых файлов и их описание (манифест)
- Пакет классов для работы java.util.jar.*
- Инструмент – архиватор jar
- Может содержать набор классов с классом, запускающим работу приложения

  java – jar nameArh.jar

Печать

- Часть графической библиотеки AWT и системы Java 2D API

Файлы в сети

- Классы URI и URL пакет java.net

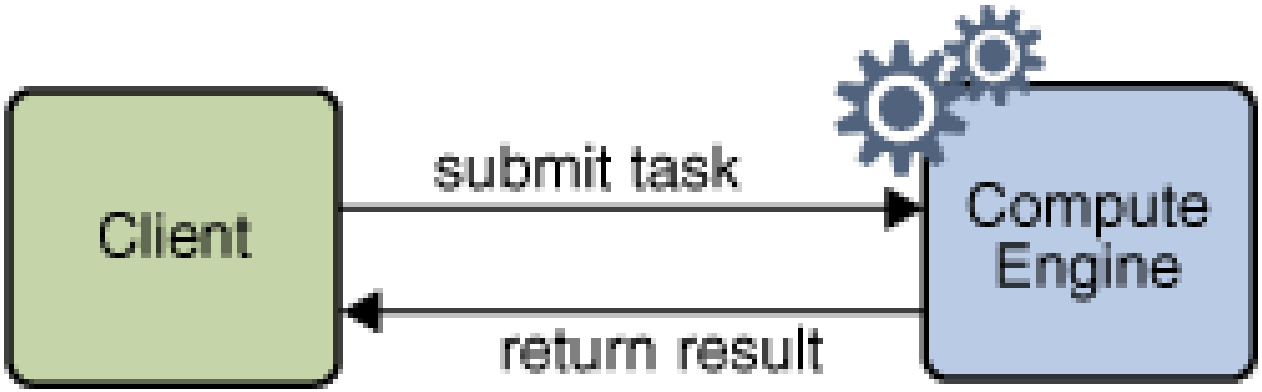
```
URL url = new URL (строка_URL);
//URLConnection hpCon = url.openConnection();
InputStream in = url.openStream();
// InputStream in = hpCon.getInputStream();
```

  – далее поток in можно обернуть фильтром, например

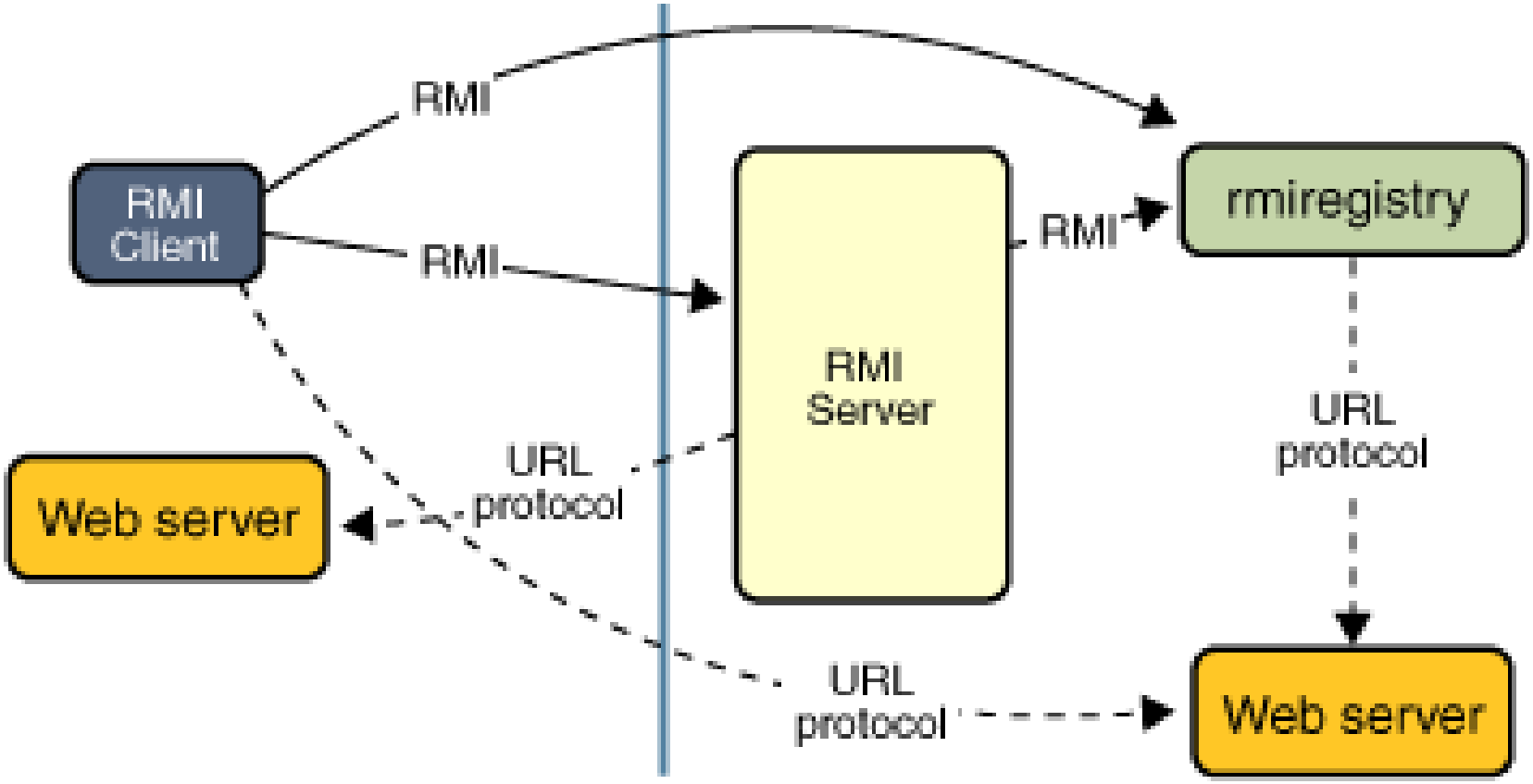
```
Scanner sin = new Scanner(in);
```

RMI

- технология распределенных приложений



Java Tutorial



Пример

```
import java.rmi.*;
/***
    Интерфейс удаленных объектов  */
public interface Product extends Remote {
        String getDescription()  throws RemoteException;
}
```

```
import java.rmi.*;
import java.rmi.server.*;
/***
  Реализация класса для удаленных объектов
*/
public class ProductImpl extends UnicastRemoteObject
                        implements Product {
   public ProductImpl(String s) throws RemoteException {
      name = s;
   }
   public String getDescription() throws RemoteException {
        return "I'm a " + name;
   }
  private String name;
}
```

```
import java.rmi.*;
import java.rmi.server.*;
import javax.naming.*;
/** Программа-сервер, регистрирует удаленный объект
    и ожидает обращения к нему со стороны клиентов */
public class ProductServer{
  public static void main(String [] args){
   try{     System.out.println ("Server start");
            ProductImpl p = new ProductImpl("JDK 1.5.0_05");
            System.out.println ("Binding...");
            Context namingContext = new InitialContext();
            namingContext.bind("rmi: Java ",p);
           System.out.println("Waiting for connect…");
   } catch(Exception e){     e.printStackTrace(); }
} }
```

```
import java.rmi.*;
import java.rmi.server.*;
import javax.naming.*;
public class ProductClient {
 public static void main(String[] args){
   System.setProperty("java.security.policy","java.policy");
   System.setSecurityManager(new RMISecurityManager());
      try{
       String url="rmi://"+args[0]+"/";
       Context namingContext = new InitialContext();
       Product c1 = (Product) namingContext.lookup(url+"Java");
    System.out.println(c1.getDescription());
  } catch(Exception e) {    e.printStackTrace(); }
}
}
```