静态代码分析指导手册

版本 1.1

IBM CDL Education Service

1 RSAR 简介与安装

1.1 RSAR 简介

RSAR (Rational Software Analyzer) 是一种可扩展的静态分析解决方案，可在开发周期的早期就开始进行软件代码评估、bug 识别和策略实施工作。随着产品在开发过程中的不断成熟，与 bug 修订和策略漏洞相关的费用将不断地增加，RSAR 恰好可以解决这一问题，因为它可以在软件开发早期识别这些问题，从而降低此类开销。RSAR 提供了企业级的可扩展框架，允许整合第三方静态分析工具，为 IT 治理、遵从性和自动化的工作流流程提供了强大的集中式报告功能，并与其他 Rational 产品进行了整合。扩展的静态分析解决方案支持软件开发生命周期早期的代码审查和错误识别。您可以从软件开发的最初阶段（即远在代码交付到代码管理和构建系统之前）运行 RSAR。RSAR 的主要特点有：

- 在软件开发生命周期早期，识别代码级问题，节省时间和金钱。
- 通过识别和解决潜在代码错误，改进整体代码质量和可预测性。
- 通过强大的报告功能，增强项目可见性并协助公司计划治理和法规遵从性。
- 通过整合到现有 Eclipse 3.3 版以上环境中，获得快速的启动和价值实现。
- 支持的操作系统：Linux、Windows

1.2 RSAR 下载和安装

1) RSAR 下载地址：
http://www14.software.ibm.com/webapp/download/search.jsp?pn=Rational+Software+Analyzer

2) 双击下载包目录中 launchpad.exe，接着 IBM Installation Manager 将会自动引导用户安装 RSAR，选择默认配置即可，单击【下一步】，直到安装完成。

2 创建 RSAR 分析器的配置

下面以创建 MyJavaApp 应用为例，演示一个 RSAR 代码分析器的作用域和规则的设置，以及运用规则进行分析并修正的全过程。

2.1 创建 Java 应用

创建一个项目：文件－>新建－>项目，输入项目名称“MyJavaApp”。

在该项目下创建包：“MyJavaApp”－>新建－>包，输入包名：“com.ibm.education”

在该包下创建类“TestRSAR.java”：“com.ibm.education”－>类，输入类名：“TestRSAR.java”。也可以直接导入源文件“TestRSAR.java”。

```
package com.ibm.education;

import java.text.SimpleDateFormat;
import java.util.Date;

public class TestRSAR {

    /**
     * This class is used to test RSAR java policy. It is a date and time format
     * application.
     * @param args
     */
    private static String defaultDatePattern = null;
    private static String defaultTimePattern = null;
    private static Date time = null;

    // ~ Methods
    //
==========================================================

    /**
     * output date and time string representing the date and time pattern
```

```
     */
    public static void main(String[] args) {
        System.out
                .println("This class is Date format application using to test RSAR
Java policy");
        time = new Date();
        System.out.println(getDate(time));
        System.out.println(getDateTime(time));
    }

    /**
     * Return default datePattern (MM/dd/yyyy)
     *
     * @return a string representing the date pattern on the UI
     */

    public static synchronized String getDatePattern() {
        defaultDatePattern = "MM/dd/yyyy";
        return defaultDatePattern;
    }

    /**
     * Return default timePattern (HH:mm)
     *
     * @return a string representing the time pattern on the UI
     */
    public static String getTimePattern() {
        defaultTimePattern = "HH:mm";
        return defaultTimePattern;
    }

    /**
     * This method attempts to return date in the form mm/dd/yyyy.
     *
```

```
 * @param aDate
 *            a date object
 * @return a formatted string representation of the date
 *
 * @see java.Util.date
 */
public static final String getDate(Date aDate) {
      SimpleDateFormat df = null;
      String returnValue = "";

      if (aDate != null) {
            df = new SimpleDateFormat(getDatePattern());
            returnValue = df.format(aDate);
      }

      return (returnValue);
}

/**
 * This method generates a string representation of a date's time in the
 * format you specify on input
 *
 * @param aDate
 *            a date object
 * @return a formatted string representation of the time
 *
 * @see java.text.SimpleDateFormat
 */
public static String getDateTime(Date aDate) {
      SimpleDateFormat df = null;     String returnValue = "";

      if (aDate == null) {
            // System.out.println("aDate is null!");
      } else {
```

```
                df = new SimpleDateFormat(getTimePattern());
                returnValue = df.format(aDate);
            }

            return (returnValue);
        }
    }
```

该类可以以“Java 应用程序”运行，输出如下结果：

This class is Date format application using to test RSAR Java policy

09/21/2009（本机实际日期）

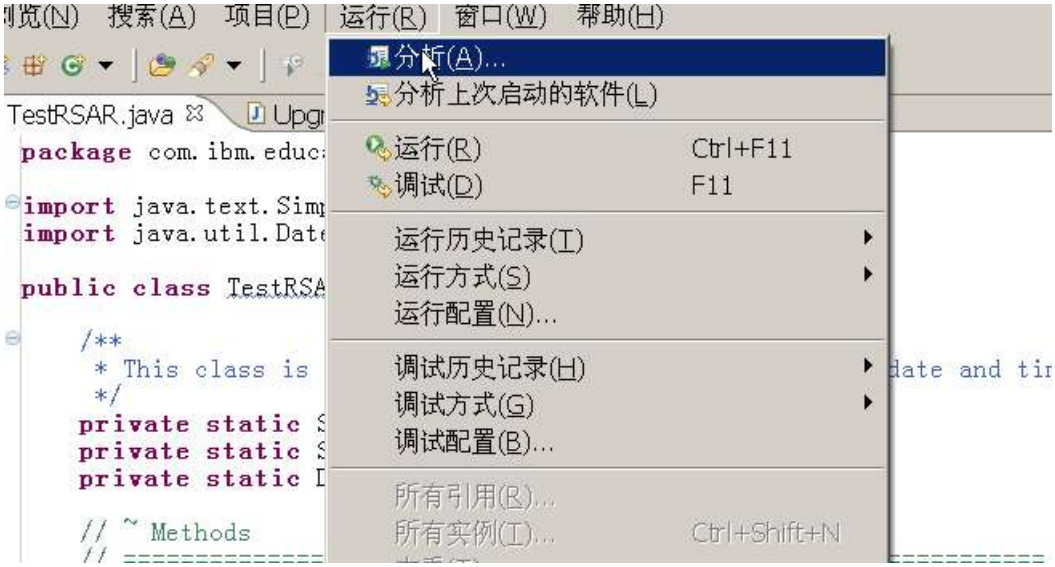
15:05（本机实际时间）

2.2 设置 RSAR 分析器的作用域

该类虽然能正常运行，但是却存在代码问题。RSAR 可帮助进行代码分析，找到并修正问题。

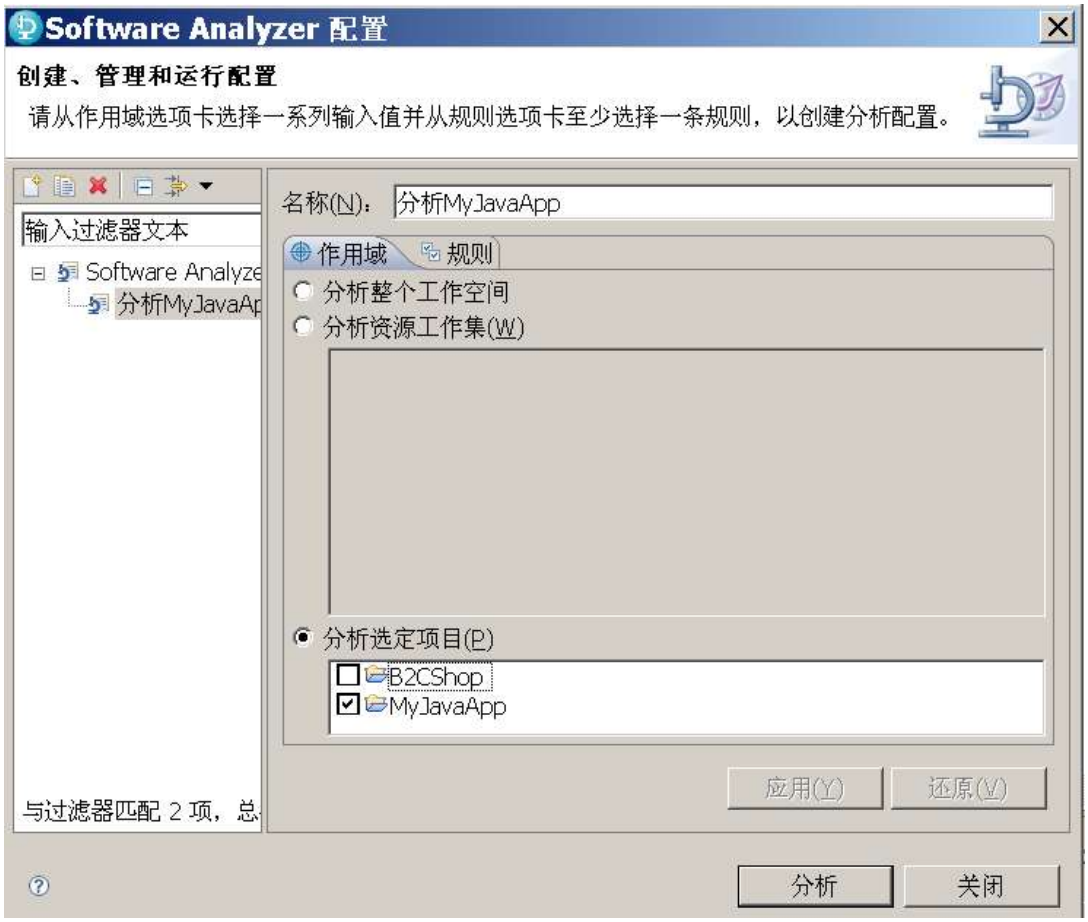
首先创建可指定分析作用域和详细信息的分析规则。

1）单击菜单栏－>运行－>分析



2） 在 RSAR 分析器的配置窗口中，创建新分析的配置。在配置名称栏中输入“分析 MyJavaApp”。

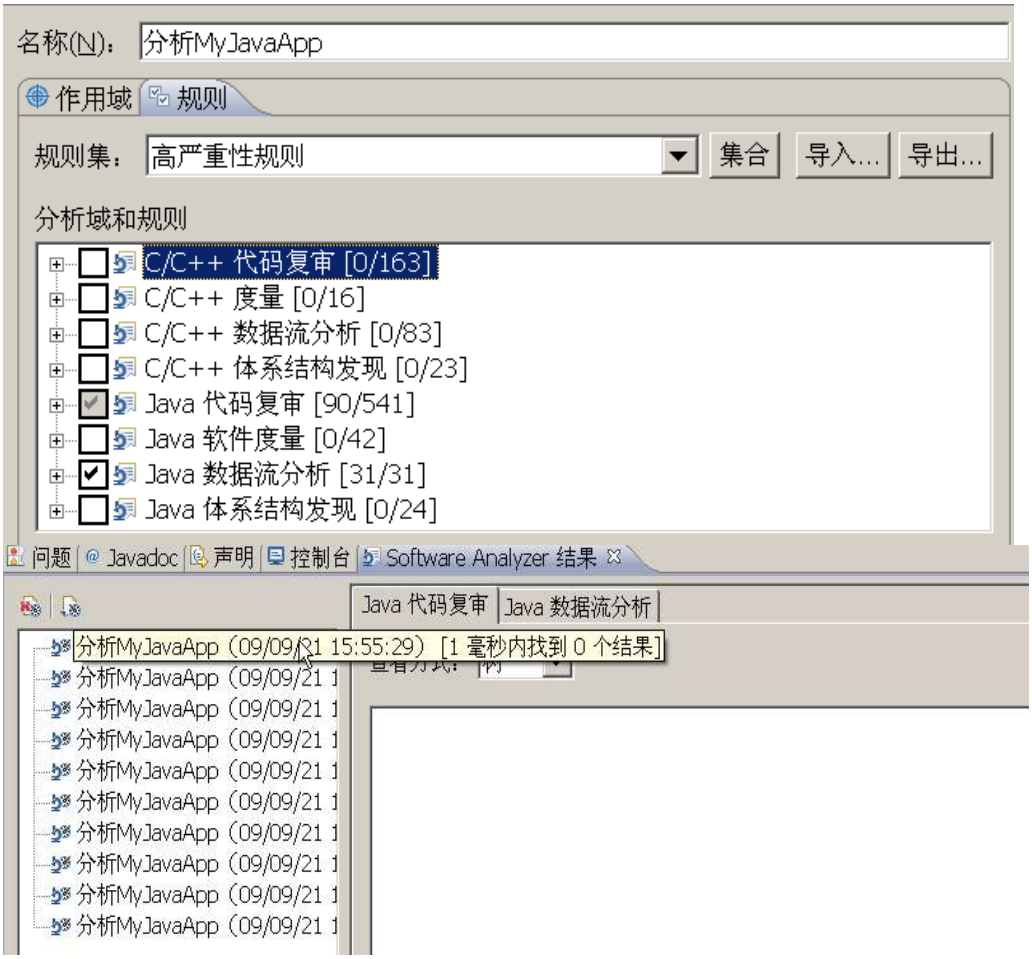
3） 选择【作用域】选项卡，点击“分析选定的项目”，并选中“MyJavaApp”。



2.3 设置 RSAR 分析器的规则

选择【规则】选项卡，选择规则集。

首先，选用“高严重性规则”规则，点击“集合”按钮引入高严重性规则。该规则将找出高严重性的代码问题。去除 C/C＋＋的选项，只保留“Java 代码复审”和“Java 数据流分析”，依次点击“应用”按钮和“分析”按钮。系统在“Software Anylyzer”视图中显示找到 0 个结果，说明该类没有高严重性的代码问题。



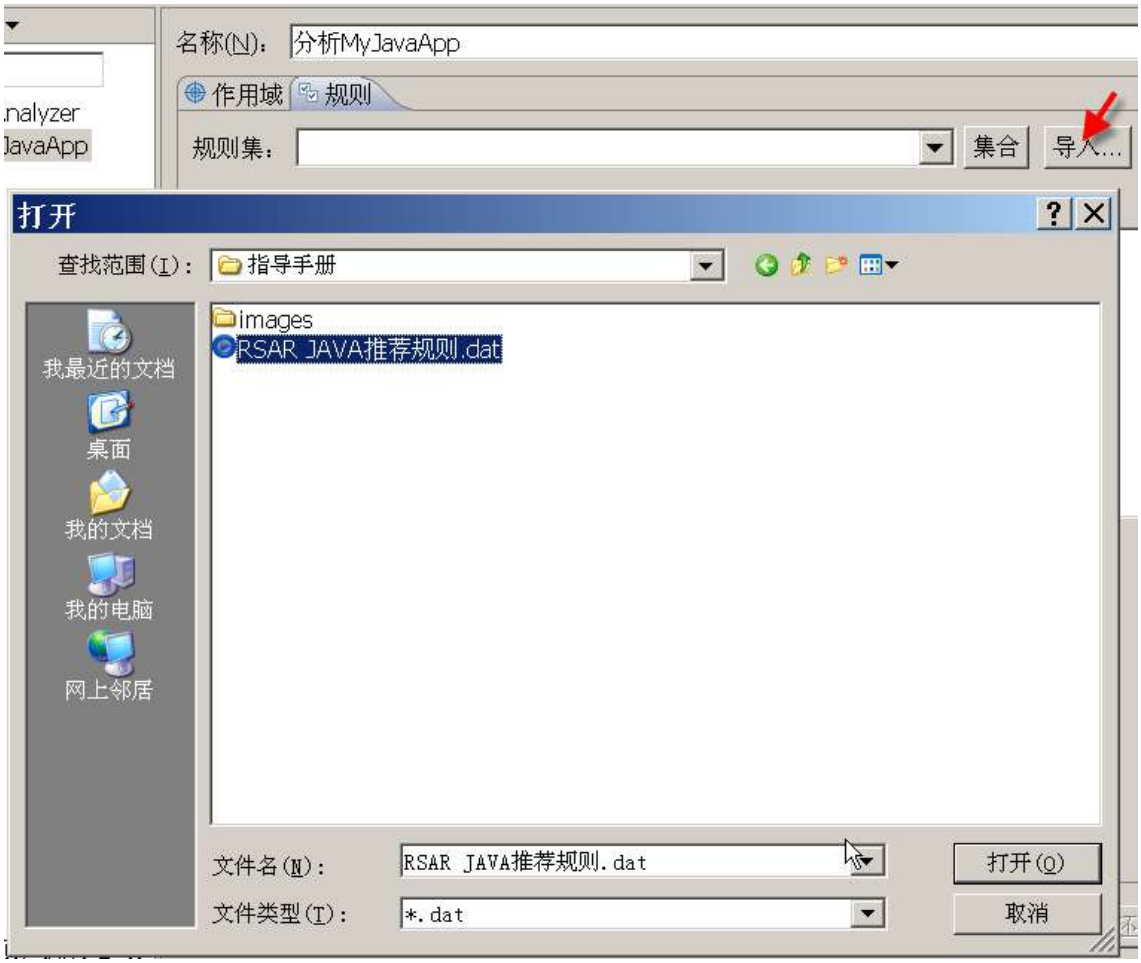
其次，选用“Java 快速代码复审”规则集合，仍然找到 0 个结果。通过了快速代码复审，说明该类符合通常意义上的 JAVA 代码要求。

再次，选用“Java 最佳实践代码复审”规则集合，找到 26 个结果。通过查看结果，发现一些分析结果属于较高要求，并不适合实训开发。

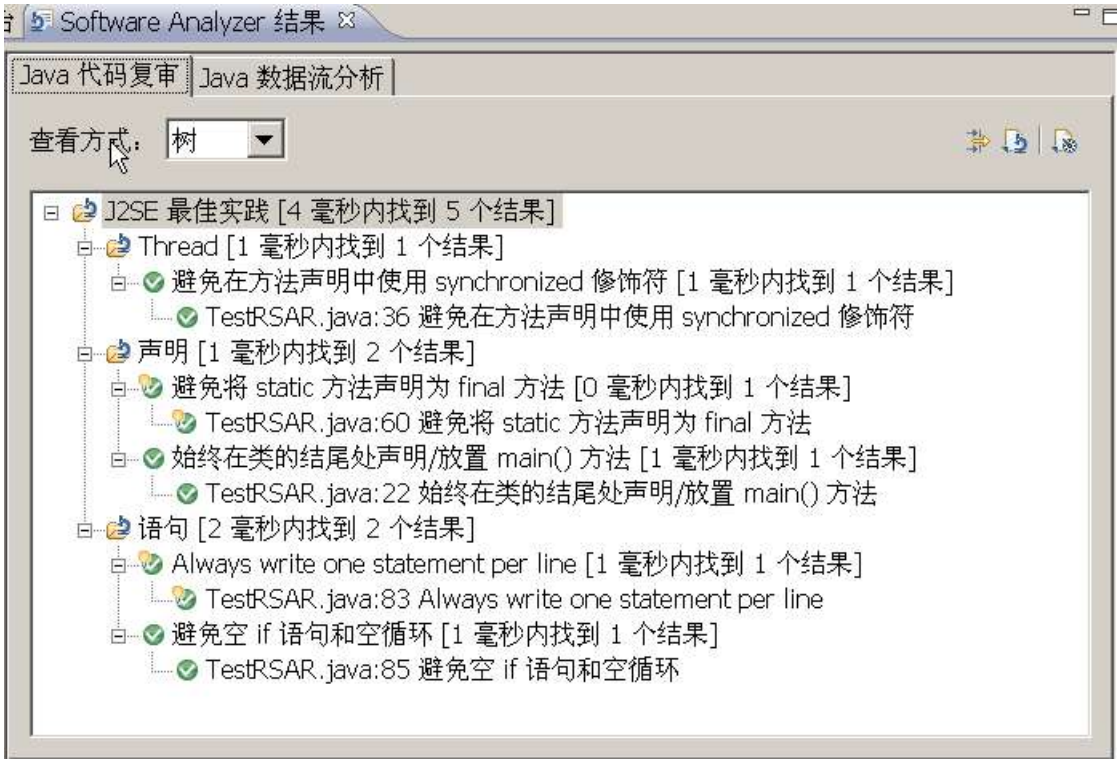
为避免过多不必要的分析结果干扰实训开发，实训定制了规则文件，大家可以导入该规则文件进行分析。该规则包含了大部分“Java 最佳实践代码复审”中与实训相关的规则，并增加了 JAVA 数据流分析。

依照下面方式导入实训提供的定制 policy 文件“RSAR JAVA 推荐规则.dat”进行代码分

析。

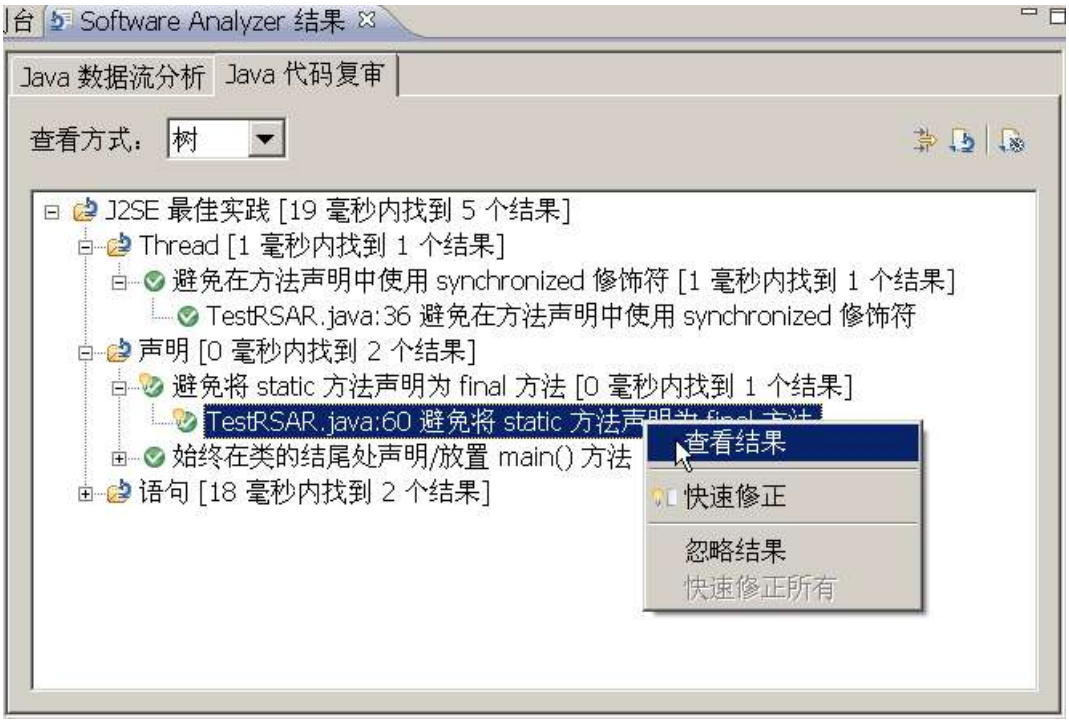


点击“应用”并“分析”，可得到如下结果：



2.4 RSAR 分析器的结果分析

点击每个结果项的右键，可以查看结果、快速修正或忽略结果。



1、第一个分析结果（提示：避免在方法声明中使用Synchronized 修饰符）的修正：

删除 synchronized 修饰符。Synchronized 会引起该方法单线程运行，除非必要，否则将影响系统性能。

```
public static synchronized String getDatePattern() {
    return defaultDatePattern;
}
```

2、第二个分析结果（避免将 static 方法声明为 final 方法）的修正：

删除 final 修饰符。Static 修饰符定义的方法已具有 final 特性。

```
public static final String getDate(Date aDate) {
        SimpleDateFormat df = null;
        String returnValue = "";

        if (aDate != null) {
            df = new SimpleDateFormat(getDatePattern());
            returnValue = df.format(aDate);
        }

        return (returnValue);
    }
```

3、第三个分析结果（始终在类的结尾处声明/放置 main()方法）的修正：

统一将 main 方法放在所在方法之后，即类的结尾处。

4、第四个分析结果（Always write one statement per line）的修正：

```
public static String getDateTime(Date aDate) {
        SimpleDateFormat df = null;    String returnValue = "";
```

将 String returnValue = ""; 换至下一行，使一行只有一个声明。

```
public static String getDateTime(Date aDate) {
        SimpleDateFormat df = null;
        String returnValue = "";
```

5、第五个分析结果（提示：避免空 if 语句和空循环）的修正：

删除“//”注释符。If 语句内必须有符合布尔条件时要运行的代码。

```
if (aDate == null) {
    // System.out.println("aDate is null!");
}
```

下面是经过修正的类。修正后再进行分析，找到 0 个结果。说明符合实训要求的规则。可导

入该类再进行运行和分析。

```
package com.ibm.education;

import java.text.SimpleDateFormat;
import java.util.Date;

public class UpgradeTestRSAR {

    /**
     * This class is used to test RSAR java policy. It is a date and time format application.
     */
    private static String defaultDatePattern = null;
    private static String defaultTimePattern = null;
    private static Date time = null;

    // ~ Methods
    // ================================================================

    /**
     * Return default datePattern (MM/dd/yyyy)
     *
     * @return a string representing the date pattern on the UI
     */

    public static String getDatePattern() {
        defaultDatePattern = "MM/dd/yyyy";
        return defaultDatePattern;
    }

    /**
     * Return default timePattern (HH:mm)
     *
     * @return a string representing the time pattern on the UI
     */
```

```
public static String getTimePattern() {
        defaultTimePattern = "HH:mm";
        return defaultTimePattern;
}

/**
 * This method attempts to return date in the form mm/dd/yyyy.
 *
 * @param aDate
 *          a date object
 * @return a formatted string representation of the date
 *
 * @see java.Util.date
 */
public static String getDate(Date aDate) {
        SimpleDateFormat df = null;
        String returnValue = "";

        if (aDate != null) {
                df = new SimpleDateFormat(getDatePattern());
                returnValue = df.format(aDate);
        }

        return (returnValue);
}

/**
 * This method generates a string representation of a date's time in the
 * format you specify on input
 *
 * @param aDate
 *          a date object
 * @return a formatted string representation of the time
 *
```

```
     * @see java.text.SimpleDateFormat
     */
    public static String getDateTime(Date aDate) {
        SimpleDateFormat df = null;
        String returnValue = "";

        if (aDate == null) {
            System.out.println("aDate is null!");
        } else {
            df = new SimpleDateFormat(getTimePattern());
            returnValue = df.format(aDate);
        }

        return (returnValue);
    }

    /**
     * output date and time string representing the date and time pattern
     * @param args
     */
    public static void main(String[] args) {
        System.out
                .println("This class is Date format application using to test RSAR Java
policy");
        time = new Date();
        System.out.println(getDate(time));
        System.out.println(getDateTime(time));
    }
}
```

2.5 JAVA 推荐规则的说明

下面对实训定制的 Java 推荐规则中的关键项进行简要说明：

1、J2SE 代码分析推荐规则：

Garbage Collection：正确使用某些常用类，避免产生内存垃圾。

Null：注意方法的返回值类型，避免某些情况下返回NULL。

String：字符串的操作很重要，有时会影响性能。String适用于长度不变的字符串处理，而StringBuffer适用于长度可变的字符串处理。

Structure：生产环境中避免使用break, continue。

Switch：恰当地使用Switch。

Thread：本实训中很少用到Thread，但需要注意慎用Synchronized。

比较：正确地比较类很重要。

初始化：养成良好的初始化习惯。

构造方法：构造方法关系对象的创建，重中之重。

集合：Set中不允许重复或空，持久化操作时建议使用Set而不是List。

可移植性：避免调用与本地绑定较紧的资源，如避免绝对路径，本机上下文环境等，以保证系统移植时的正确性。

声明：声明非常重要，在最佳实践中有些较高，建议实训中仅选择推荐的规则文件中的分析项即可。

数据类型转换：开发中数据类型转换很常见，请注意必要的cast。

条件：条件操作很常见，易出现逻辑失误。

序列化：调用非本地JVM的对象时须进行序列化，推荐JavaBean序列化。

循环：生产环境中，循环使用不当，很易出现问题。

异常：生产环境中，必须正确处理异常。

语句：符合JAVA的编程规范。

2．J2EE代码分析推荐规则：

推荐使用“Java最佳实践代码复审”规则集合来进行代码分析。各小组开发经理也可以根据组内代码实际情况进行定制。

在导入“Java最佳实践代码复审”规则集合后，选择“Java代码复审”－>“J2EE最佳实践”，推荐选择下列几个主要的关键项：

Servlet：Servlet是多线程运行的，尽量避免使用单线程的特性。

垃圾回收：如果重写Servlet的finalize()方法，一定注意不要调用某些影响垃圾回收的类。

数据竞争：Servlet是多线程的，尽量避免在Servlet中调用GUI方面的类。

性能和可伸缩性：在Servlet中调用某些类，会影响性能和系统的灵活性，如DriverManager连接数据库的方式，会降低性能，应避免使用。

正确性：应注意如存储在HttpSession中的对象需要序列化。

请注意，由于JSP运行时是编译成Servlet运行的，所以以上对Servlet的代码分析项，同样适用于JSP。

另外，“JAVA 数据流分析”－>“资源泄漏”，对实训开发也很有意义。

推荐选择下列几个主要的关键项：

J2EE：正确处理 HttpSession 对象。

SQL：对数据库操作的代码质量对性能影响较大，操作数据库后正确关闭链接非常重要。

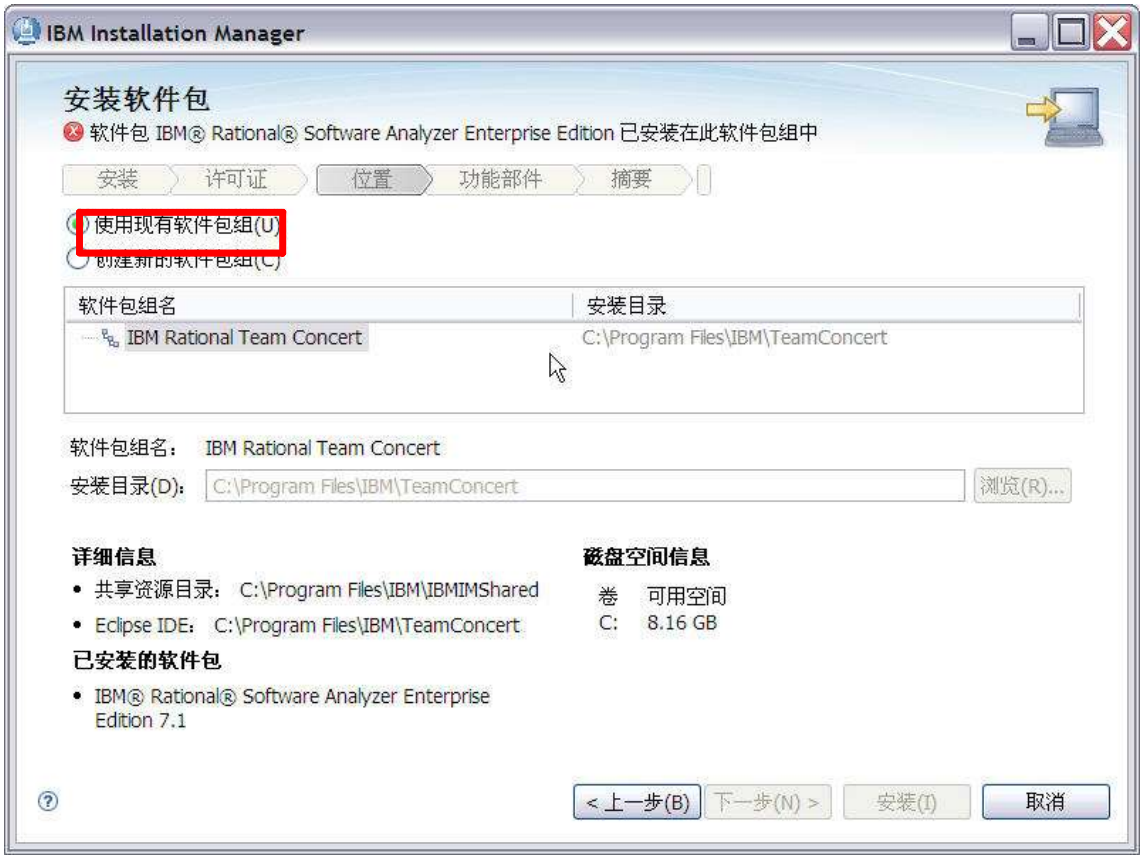
流：正确关闭流。

3 RSAR 与 RTC 集成

RTC (Rational Team Concert) 是 Jazz 平台的基于 Eclipse RCP 的客户端，是一个为软件开发团队创造协同工作环境的软件。您可以选择想要应用到所有代码交付的 RSAR 规则进行分析。安装 RSAR 时，您必须扩展现有 RTC Eclipse IDE 才能使用此功能。与 RTC 的集成将在您的存储库项目区域设置中进行。您必须拥有对项目区域设置进行编辑的适当权限，才能完成此任务。如果照此在项目区域中启用了 RSAR 规则，那么每次团队成员尝试交付文件时，都将使用选定的规则来分析要交付的文件。如果文件违反了选定的规则，那么交付操作将失败且【团队顾问程序】视图将打开，这表示静态分析已导致违例。要能够交付文件，首先您必须修正代码，以便它不会中断静态分析规则。

使用 RTC 进行源代码控制，那么可以使用 RSAR 对其进行扩展，并将项目区域设置为无论何时向团队存储库交付文件，都包含 RSAR 规则检查。启用此设置后，仅当文件不包含已配置的 RSAR 规则违例时，才能交付到工作空间流。当 RSAR 与 RTC 集成后，您也可以从前者的结果直接创建后者的工作项。

如果已安装好 RTC，当安装 RSAR 时候，请在 IBM Installation Manager 引导安装过程，将默认的安装软件包选项更改【使用现有软件包组】。

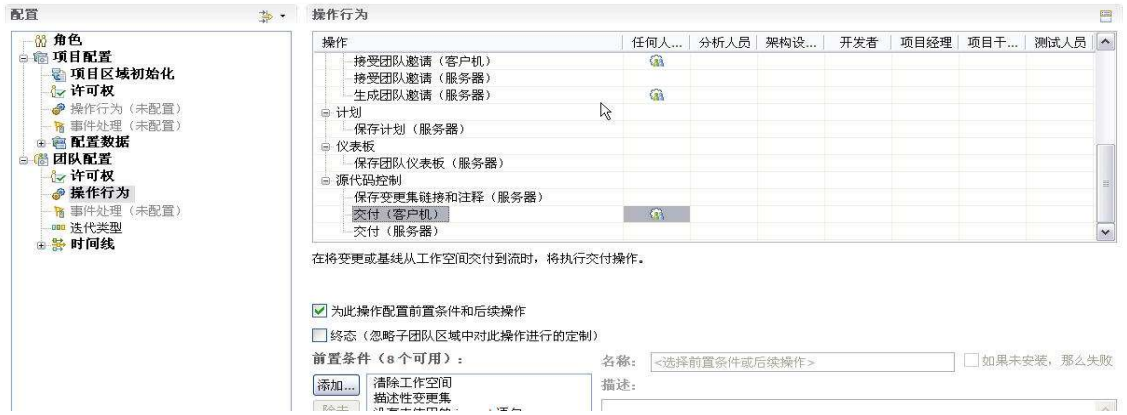


接着在 RTC 环境中，安装 RSAR:

1) 单击【过程模板】视图中的模板。

2) 单击【打开】，【过程模板】会在编辑器视图中打开，单击【过程配置】选项卡，展开【团队配置】部分，然后单击【操作行为】部分。

3) 在【操作列表】中，选择【源代码控制】->【交付（客户机）】，然后选择【每个成员（缺省）】列表或列中的图标( )。



4) 在【前置条件】列表中，单击【添加】。

5) 在【添加前置条件】对话框中，请选择【Rational Software Analyzer】，然后单击【确定】。

在前置条件列表中，请确保已选定 Rational Software Analyzer。这将导致使用特定于 Rational Software Analyzer 的设置来填充前置条件描述区域。

6) 在前置条件描述区域规则列表中，选择想要应用到所有代码交付的 RSAR 规则。

7) 【应用变更】对项目区域、团队区域或模板所做的更改。