Prior to subscribing for, installing or using any Third Party Products and Services it is your responsibility to ensure that your airtime service provider has agreed to support all of their features. Some airtime service providers may not offer Internet browsing functionality with a subscription to BlackBerry® Internet Service. Check with your service provider for availability, roaming arrangements, service plans and features. Installation or use of Third Party Products and Services with RIM's products and services may require one or more patent, trademark, copyright or other licenses in order to avoid infringement or violation of third party rights. You are solely responsible for determining whether to use, Third Party Products and Services and if any third party licenses are required to do so. If required you are responsible for acquiring them. You should not install or use Third Party Products and Services until all necessary licenses have been acquired. Any Third Party Products and Services that are provided with RIM's products and services are provided as a convenience to you and are provided "AS IS" with no express or implied conditions, endorsements, guarantees, representations or warranties of any kind by RIM and RIM assumes no liability whatsoever, in relation thereto. Your use of Third Party Products and Services shall be governed by and subject to you agreeing to the terms of separate licenses and other agreements applicable thereto with third parties, except to the extent expressly covered by a license or other agreement with RIM.

Certain features outlined in this documentation require a minimum version of BlackBerry® Enterprise Server, BlackBerry® Desktop Software, BlackBerry® Device Software and/or BlackBerry Handheld Software and may require additional development or Third Party Products and Services for access to corporate applications.

Certain products mentioned in this documentation include software developed by the Apache Software Foundation (http://www.apache.org/) and/or licensed pursuant to Apache License, Version 2.0 (http://www.apache.org/licenses/). For more information, see the NOTICE.txt file included with the software. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

The terms of use of any RIM product or service are set out in a separate license or other agreement with RIM applicable thereto. NOTHING IN THIS DOCUMENTATION IS INTENDED TO SUPERSEDE ANY EXPRESS WRITTEN AGREEMENTS OR WARRANTIES PROVIDED BY RIM FOR PORTIONS OF ANY RIM PRODUCT OR SERVICE OTHER THAN THIS DOCUMENTATION.
Chapter 7
Testing and debugging an application

Objectives

• Explain the importance of preverifying applications
• Identify the available test tools
• Explain the available debugging tools in BlackBerry® JDE Plug-in for Eclipse®
• Explain garbage collection, memory leaks, deadlocks and their importance
• Explain the Event Log

This chapter outlines the various tools that you can use to verify that applications you develop for the BlackBerry® smartphone function properly.
When you **preverify** classes, you reduce the amount of processing that the BlackBerry smartphone must perform when you install your BlackBerry Java® application. To partially verify your **classes** manually before you install your application on a Blackberry smartphone, you can use the Preverify tool, available with the BlackBerry® Java® Development Environment.

> Issue a command from the command line in the following format:

```
preverify.exe [-d] output -classpath directory input; directory
```

Integrated Development Environments preveryify your classes automatically.
Methods for testing applications

This chapter discusses the following methods for testing BlackBerry smartphone applications:

- testing applications on the BlackBerry Smartphone Simulator
- testing applications on the Blackberry smartphone

Testing applications on a BlackBerry Smartphone Simulator

The BlackBerry Smartphone Simulator enables you to run BlackBerry smartphone applications on your computer. The BlackBerry Smartphone Simulator includes the BlackBerry smartphone applications that are typically available on BlackBerry smartphones and enables you to load and test your own applications. You can simulate and test various connectivity and state changes using the BlackBerry Smartphone Simulator.

The BlackBerry Smartphone Simulator runs the same Java code that the Blackberry smartphone runs, so the BlackBerry Smartphone Simulator provides an accurate environment for testing how applications function on a Blackberry smartphone. The BlackBerry® Java® Development Environment includes current versions of the BlackBerry Smartphone Simulator.

For more information about using the BlackBerry Smartphone Simulator, see The BlackBerry Smartphone Simulator Version 4.7 Development Guide.

Testing applications on a Blackberry smartphone

After you test your application on the BlackBerry Smartphone Simulator, you can install your application on a Blackberry smartphone. If your application uses signed APIs, you require code signing keys. After you install the application on the Blackberry smartphone, you can open the application and test its functionality and performance. For debugging purposes, you can attach your device to the BlackBerry® Integrated Development Environment and use the debugging tool to step through your application code. The BlackBerry IDE can be useful if you are trying to identify a network or Bluetooth® issue, or other issues that are difficult to simulate.

The following screen shows the Debug Configurations tool interface that allows you to specify where you connect to when debugging.
Figure 9.1 Debug Configurations tool interface for transferring applications to a BlackBerry smartphone

1. Under **Debug Configurations** click **BlackBerry Device** and then **New configuration**.

2. Select the option to attach the debugger to the specific device.

3. Enter the device password if prompted.

4. If the error message that one or more .debug files are missing, the software version on your device does not match the software version of your simulator. For more information about the BlackBerry Smartphone Simulator, visit http://na.blackberry.com/eng/developers/resources/simulators.jsp.

The following screen shows the missing files error message.
If the simulator is not available for your version of device software, then install the BlackBerry Smartphone Simulator that matches the software version on your BlackBerry smartphone.

**Note:**
You can also ignore the message, and click the Don’t ask this again button. When you ignore the missing files message, some of the debugging features are not present, such as getting stack traces, but most features are available.

*Figure 9.2 Missing file warning*
1. You can test your application on
   A. On the BlackBerry simulator
   B. In the BlackBerry smartphone
   C. In the Visual Studio environment
   D. Both B and C

2. The BlackBerry simulator
   A. Does not include applications already available on the BlackBerry smartphone
   B. Includes applications found on the BlackBerry smartphone
   C. Does not support any connectivity options
   D. Is not included with BlackBerry Java Development Environment

3. Once installed on a smartphone, your application
   A. Can not be tested
   B. Can be tested using the BlackBerry IDE
   C. Can be tested only if have installed it with code signing keys
   D. Can't be used to test network connectivity issues

4. What does the message indicating that a debug file is missing indicate?
   A. You are attempting to debug on the BlackBerry simulator
   B. You are attempting to debug outside of the BlackBerry IDE
   C. Your simulator and the BlackBerry IDE are different versions
   D. You are not in debug mode yet.

5. If you are missing debug files you can?
   A. Upgrade your smartphone
   B. Down grade your smartphone
C. Use a different version of the simulator
D. Any of the above
Answers

1. D
2. B
3. B
4. C
5. C
Debugging tools available in BlackBerry JDE
Plug-in for Eclipse

When you debug an application, you search for and troubleshoot errors in your application. Multiple debugging methods are available. You can run the application on the BlackBerry Smartphone Simulator or on the device.

In most situations, you debug applications on the BlackBerry Smartphone Simulator as it provides better tools for identifying issues. You can identify certain issues more easily by debugging on the device; for example: making network connections through a wireless service provider WAP gateway.

Open applications in Eclipse with the following methods:
- click Run/Run option (Ctrl – F11)
- click Run/Debug (F11)

To debug applications in Eclipse®, you must click Run/Debug.

![Image of Eclipse IDE with debug settings open]
BlackBerry JDE Plug-in for Eclipse allows you to modify the workspace layout. The following figure displays a typical layout. In the top right corner, the Java button is selected. Beside the Java button is the Debug button, which switches the view into the Debug view.

This screen shows the Debug view.

- If these buttons are not available you can find the options in the Window menu, and then select Show View/Debug. The Debug tab (top left) displays the running threads.
- If the simulator is not running most of the windows are empty.
- The section below the debug layout displays the source code. You can add breakpoints in this section. You can add breakpoints before running the simulator or while running it. The simulator stops application execution when it hits the breakpoint.
- Below the source code window is the Console window. The Console window displays the output from the virtual machine, as well as the messages from your own application.
- The top right corner displays the list of variables and values, as well as a list of breakpoints.
Console window

You configure the Console window to display custom messages in addition to the messages from the BlackBerry® Java® Virtual Machine.

If you use the System.out.println("message") command, or System.err.println("message"); the text message or value of x appears in the Console window.

You can enter try-catch blocks to catch exceptions, and then print as much information as required.

```java
try {
    ...
}

catch (Exception e) {
    System.out.println("Error 123");
    System.out.println(e.toString());
    e.printStackTrace();
}
```

The above commands print your custom error number, exception name, and a Stack if available. Add this sort of message to the event log.

Variables and Debug window

After the application hits the breakpoint and stops, the local variables appear in the Variables and Debug window.

This screen shows the Variables view.
Figure 9.5 Variables view

The Variables window displays the values of all local variables at the breakpoint.

Figure 9.6 Debug window

On the top, right part of the Debug window there are control icons. The control icons allow you to resume your application, step in to, and step over the code. You can also use the following function keys:

- resume application - (F8)
- step in to the code (F5)
- step over the code (F6)
This displays the step-by-step functioning of the application.

The Debug window also displays the running threads. You can pause and resume the application manually, and setup and apply step filters.

**BlackBerry Memory Statistics View**

To see the *BlackBerry Memory Statistics View*, as well as the BlackBerry Objects View and BlackBerry Profiler View, click Window > Show View > Other > expand BlackBerry, and then set the view you want. This screen shows the BlackBerry Memory Statistics View.

![Figure 9.7 View select](image)

To use the memory statistics view, use the following procedure:

1. Set up two breakpoints.

2. When the application stops at the first breakpoint, click **Refresh** on the BlackBerry Memory Statistics View.

3. Click **Save** to save the data in .csv format.
At this point, you are capturing a snapshot to use for a comparison.

This screen shows the BlackBerry Memory Statistics View.

![Image of BlackBerry Memory Statistics View](image)

**Figure 9.8** BlackBerry Memory Statistics View

4. Press F8 to go to the next breakpoint.

5. When the application stops, refresh the BlackBerry Memory Statistics View again.

6. Click **Compare**. The system displays the changes.

The next screen shows the BlackBerry Memory Statistics View with object handles used. An excessive amount of RAM is used as well. You can investigate the cause. In this case, bitmap images are loaded in the application.
BlackBerry Objects View

The **BlackBerry Objects View** allows you to compare sections of application code between breakpoints. You must add breakpoints to your code for this feature to function properly. After the application reaches the breakpoint, refresh the BlackBerry Objects View.
The BlackBerry Objects View displays all objects from all running applications.

The Save option saves the data in .csv format. You can use the Snapshot option to compare the objects between two breakpoints, and is useful for finding memory leaks.

You can also run garbage collection to see if object references are properly deallocated.

This screen shows the BlackBerry Objects View Options.
Use the Snapshot Filter to limit the BlackBerry Object View to your process. In the example, the process number is 121. After selecting the required option in the **Snapshot Filter** drop-down list, enter the process number, and then click **Ok**. You must refresh the main window to see the update.

The list of all the objects in our application appear and you can monitor and analyze the memory usage.

**Figure 9.12** All objects view

**BlackBerry Profiler View**

The *BlackBerry Profiler View* displays information about application states. The summary view displays percentage spent in Idle, Code Execution and Garbage Collection stages.
The Method view displays a detailed view of all running packages. In the Options menu, choose what package to profile. Default settings are in ticks (time), but you can also view size and number of objects.

Do not create too many String objects.

```java
String test = "a";
test = "b";
```
test += "c";
System.out.println(test.concat("d"));

The preceding code does have one variable test but it did not create only one String object. Run the profiler to see how many String objects this code generates.
1. To add breakpoints to your code, you should use the ________ window
   A. Console
   B. Source
   C. Variable list
   D. Tasks

2. The Console window will display
   A. Only your messages
   B. Only messages from the Java Virtual Machine
   C. Your messages and messages from the JVM
   D. No messages

3. When are local variables displayed?
   A. Constantly
   B. When the application hits a breakpoint
   C. When the application starts
   D. When the debugger is refreshed during an exception

4. The difference between pressing F5 and F6?
   A. F6 will step into any methods being called from the current line
   B. F5 will step into any methods being called from the current line
   C. F5 will cause the application to run to its end or to the next breakpoint
   D. F6 will cause the application to run to its end or to the next breakpoint

5. The memory statistics View is
   A. Always displayed
   B. Displayed whenever the application stops on a breakpoint
C. Hidden during exceptions
D. Displayed when you select it through the Window menu
Answers

1. B
2. C
3. B
4. A
5. D
Garbage collection, memory leaks and deadlocks

This section discusses how garbage collection, memory leaks and deadlocks are relevant to the BlackBerry Java Development Environment.

Garbage collection

The BlackBerry Smartphone is a purely Java based platform and it implements a system called garbage collection to periodically free its memory and resources. All the objects that do not have a reference remain in memory until a garbage collection runs and frees that memory.

If you create a String object String test = “some text”; it occupies some space in the memory. If you do not need that object, you can indicate test = null; however, the memory is still occupied with some text. Only when a garbage collection runs is that space in memory freed.

You can run garbage collection manually; however, doing so can affect running BlackBerry applications. When you run garbage collection on a BlackBerry Smartphone, no other process runs.

<table>
<thead>
<tr>
<th>GC type</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM only</td>
<td>0.5 seconds</td>
</tr>
<tr>
<td>Flash + RAM (full)</td>
<td>1 second</td>
</tr>
<tr>
<td>Persistent</td>
<td>10 seconds</td>
</tr>
<tr>
<td>Emergency</td>
<td>20 + seconds</td>
</tr>
<tr>
<td>Thorough</td>
<td>20 to 30 seconds</td>
</tr>
</tbody>
</table>

The preceding table displays the garbage collection types and the approximate time it takes for them to run on the BlackBerry Smartphone. If the system runs a garbage collection option other than RAM or Full, it has a significant impact on the user experience.

Even quick garbage collection types affect applications that require fast responses, such as arcade games.

The BlackBerry heuristics monitor checks for available free memory and runs garbage collection as needed. It runs quickly and not very often.

Other applications that run in the background can affect applications. Therefore, it is important to reduce memory usage and garbage collection usage.
Memory leaks

Memory leaks are created when a reference to an unneeded object is maintained. Even if you design code that deletes these references, a reference remains in the system that prevents that object from being deleted.

Memory leaks can happen anywhere in the code but are usually found in the following:

- Data Structure
- Local Variables
- Runtime Store
- Listeners

Memory leaks are not easy to detect, but the following symptoms suggest memory leaks:

- hourglass appears often as the device is trying to do garbage collection
- email messages start being deleted

When the BlackBerry Smartphone is running out of memory it attempts to notify all applications that use the low memory manager and ask them to free some space. The email application then deletes some messages to try to create space.

To view the free object handles, go to Options/Status on your BlackBerry Smartphone. The File Free number appears. If the number is low, it can indicate a memory leak.

Note:

One method for finding memory leaks is to make the leak worse. The more data the application uses, the more it leaks and it can be easier to detect.

Deadlocks

Deadlocks occur when two or more threads are waiting for each other, and therefore the application becomes blocked. The BlackBerry JVM detects that the application is not responsive and terminates the application eventually. On the BlackBerry Smartphone, you can detect and prevent deadlocks.
Figure 9.15 Selecting the deadlock debugger

1. Under BlackBerry Simulator, select **DebugServer**.

2. Select the required Debug configurations.

3. On the **Debugging** tab, select the **Interrupt debugger on potential deadlock** check box.

4. Click **Run**.

If the simulator detects a deadlock or a potential deadlock it stops the execution of the application and displays the details in the Console window.
1. Memory is freed only when
   A. A variable is set to null
   B. Garbage collection is run
   C. Neither A or B
   D. Both A and B

2. When garbage collection runs
   A. All applications pause
   B. The current application pauses
   C. Background applications pause
   D. Foreground applications pause

3. The term "memory leak" refers to
   A. Having an unused reference to a maintained object
   B. Maintaining a reference to an unused object
   C. Forgetting things
   D. MicroSD cards falling out of the smartphone

4. A sign of memory leaks is
   A. Screen flicker
   B. High Free File number
   C. Frequent garbage collection
   D. Infrequent garbage collection

5. When the JVM detects a deadlock, it
   A. Eventually terminates the application
   B. Immediately terminates the application
Testing and debugging an application

C. Refreshes the application
D. Does nothing

**Answers**

1. D
2. B
3. B
4. C
5. A
Event Log

The event log is the log of system events. You can write to the event log using the available APIs. The event log is a useful tool to debug your application. You can access the event log on the BlackBerry Smartphone or copy it to your computer using the application called Javaloader.

To access the event log on the BlackBerry Smartphone (and on the simulator), perform the following procedure:

1. From the Home screen, hold down the Alt key, and then type 1g1g.

A list of all logged events appears on your Home screen.

![Logged events](image)

*Figure 9.16* Logged events

2. Press Enter to see more details about every item in the log.
3. To download the event log from the device to the PC type in the command prompt.

java -loader -u eventlog log.txt

The API which allows your application to write to the event log is as follows.

\begin{verbatim}
net.rim.device.api.system.EventLogger

For example

// Register application for event logging.
EventLogger.register(0x9c805919833654d6L, SampleApp);

// Set minimum logging level.
EventLogger.setMinimumLevel(EventLogger.INFORMATION);

// Log a numeric event.
EventLogger.logEvent(0x9c805919833654d6L, 12, EventLogger.INFORMATION);
\end{verbatim}

\textbf{Note:}\n
Add entries to the event log sparingly. The event log has a limited size so new entries overwrite older entries. Writing to this log also increases overhead.
// Log a String

EventLogger.logEvent( GUID, yourString.getBytes(), level );
Testing and debugging an application

This chapter describes various methods for testing and debugging BlackBerry Java applications.

When you preverify your classes, you reduce the amount of processing that the BlackBerry® Smartphone must perform when you install your BlackBerry Java application.

You can test applications in the BlackBerry Smartphone Simulator or on the BlackBerry Smartphone. The BlackBerry Smartphone Simulator runs the same Java code as the BlackBerry smartphone, so the BlackBerry Smartphone Simulator provides an accurate environment for testing how applications functions on a BlackBerry smartphone. You can test on a BlackBerry Smartphone to better verify functions such as network connections.

BlackBerry JDE Plug-in for Eclipse allows you to debug an application with the following interfaces:
- console window, which displays the output from the virtual machine, as well as the messages from your own application.
- variable and debug window, which displays the values of all local variables at the breakpoint
- BlackBerry Memory Statistics View, which displays amounts of RAM used by an application
- BlackBerry Objects View, which displays all objects, from running applications to view memory usage
- BlackBerry Profiler View, which displays information about application states

Garbage collection periodically frees memory and resources. All the objects that do not have a reference remain in memory until a garbage collection runs and frees that memory.

Memory leaks are created when a reference to an unneeded object is maintained. Even if you design code that deletes these references, a reference remains in the system that still prevents that object from being deleted.

Deadlocks occur when two or more threads are waiting for each other, and therefore the application becomes blocked.

The event log is a log of all system events. You can write to the event log using available APIs.
Review Questions

1. Describe the options available for testing your smartphone application including what can be best tested on each device.

2. Describe the purpose of these windows when debugging:
   - Console:
   - Variable list:
   - Source:
   - Debug:

4. What are the steps required to compare memory statistics for two points in your application?

5. Describe the role of garbage collection and what happens when the smartphone starts running out of memory.

6. Describe the steps for detecting deadlocks during debugging.

7. When debugging, an alternative to writing to the console window is to
   - E. Send e-mail messages
   - F. Write to the event log
   - G. Write to the source window
   - H. Display notification dialogs in the IDE