SELF TEST ANSWERS

Understanding Packages

1. Which two `import` statements will allow for the import of the `HashMap` class?

   A. `import java.util.HashMap;`

   B. `import java.util.*;`

   C. `import java.util.HashMap.*;`

   D. `import java.util.hashMap;`

Answer:

☑ **A** and **B**. The `HashMap` class can be imported directly via `import java.util.HashMap` or with a wild card via `import java.util.*;`.

☒ **C** and **D** are incorrect. **C** is incorrect because the answer is a static `import` statement that imports static members of the `HashMap` class, and not the class itself. **D** is incorrect because class names are case-sensitive, so the class name `hashMap` does not equate to `HashMap`.

2. Which statement would designate that your file belongs in the package `com.scjaexam.utilities`?

   A. `pack com.scjaexam.utilities;`

   B. `package com.scjaexam.utilities.*`

   C. `package com.scjaexam.utilities.*;`

   D. `package com.scjaexam.utilities;`

Answer:

☑ **D.** The keyword `package` is appropriately used, followed by the package name delimited with periods and followed by a semicolon.

☒ **A, B,** and **C** are incorrect answers. **A** is incorrect because the word `pack` is not a valid keyword. **B** is incorrect because a `package` statement must end with a semicolon. **C** is incorrect because you cannot use asterisks in `package` statements.

3. Which of the following is the only Java package that is imported by default?
   A. `java.awt`
   B. `java.lang`
   C. `java.util`
   D. `java.io`

Answer:

☑ **B.** The `java.lang` package is the only package that has all of its classes imported by default.

☒ **A, C,** and **D** are incorrect. The classes of packages `java.awt`, `java.util`, and `java.io` are not imported by default.

4. What Java-related features are new to J2SE 5.0?
   A. Static imports
   B. `Package` and `import` statements
   C. Autoboxing and unboxing
   D. The enhanced for-loop

Answer:

☑ **A, C,** and **D**. Static imports, autoboxing/unboxing, and the enhanced `for`-loop are all new features of J2SE 5.0.

☒ **B** is incorrect because basic `package` and `import` statements are not new to J2SE 5.0.

Understanding Package-Derived Classes

5. The `JCheckBox` and `JComboBox` classes belong to which package?
   A. `java.awt`
   B. `javax.awt`
   C. `java.swing`
   D. `javax.swing`

Answer:

☑ **D.** Components belonging to the Swing API are generally prefaced with a capital J. Therefore, `JCheckBox` and `JComboBox` would be part of the Java Swing API and not the Java AWT API. The Java Swing API base package is `javax.swing`.

☒ **A, B,** and **C** are incorrect. **A** is incorrect because the package `java.awt` does not include the `JCheckBox` and `JComboBox` classes since they belong to the Java Swing API. Note that the package `java.awt` includes the `CheckBox` class, as opposed to the `JCheckBox` class. **B** and **C** are incorrect because the package names `javax.awt` and `java.swing` do not exist.

6. Which package contains the Java Collections Framework?

A. `java.io`

B. `java.net`

C. `java.util`

D. `java.utils`

Answer:

☑ **C.** The Java Collections Framework is part of the Java Utilities API in the `java.util` package.

☒ **A, B,** and **D** are incorrect. **A** is incorrect because the Java Basic I/O API's base package is named `java.io` and does not contain the Java Collections Framework. **B** is incorrect because the Java Networking API's base package is named `java.net` and also does not contain the Collections Framework. **D** is incorrect because there is no package named `java.utils`.

7. The Java Basic I/O API contains what types of classes and interfaces?

A. Internationalization

B. RMI, JDBC, and JNDI

C. Data streams, serialization, and file system

D. Collection API and data streams

Answer:

☑ **C.** The Java Basic I/O API contains classes and interfaces for data streams, serialization, and the file system.

☒ **A, B,** and **D** are incorrect because internationalization (i18n), RMI, JDBC, JNDI, and the Collections Framework are not included in the Basic I/O API.

8. Which API provides a lightweight solution for GUI components?

A. AWT

B. Abstract Window Toolkit

C. Swing

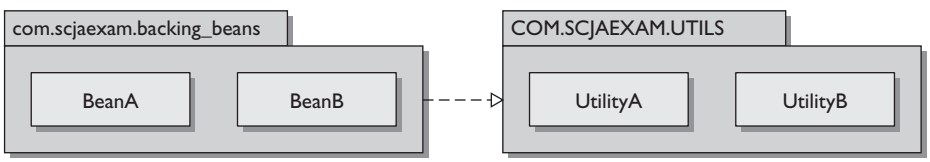
D. AWT and Swing

Answer:

☑ **C.** The Swing API provides a lightweight solution for GUI components, meaning that the Swing API's classes are built from pure Java code.

☒ **A, B,** and **D** are incorrect. AWT and the Abstract Window Toolkit are one and the same and provide a heavyweight solution for GUI components.

9. Consider the following illustration. What problem exists with the packaging? You may wish to reference Chapter 9 for assistance with UML.



A. You can only have one class per package.

B. Packages cannot have associations between them.

C. Package `com.scjaexam.backing_beans` fails to meet the appropriate packaging naming conventions.

D. Package `COM.SCJAEXAM.UTILS` fails to meet the appropriate packaging naming conventions.

Answer:

☑ **D.** `COM.SCJAEXAM.UTILS` fails to meet the appropriate packaging naming conventions. Package names should be lowercase. Note that package names should also have an underscore between words; however, the words in "scjaexam" are joined in the URL, therefore excluding the underscore here is acceptable. The package name should read `com.scjaexam.utils`.

☒ **A, B,** and **C** are incorrect. **A** is incorrect because being restricted to having one class in a package is ludicrous. There is no limit. **B** is incorrect because packages can and frequently do have associations with other packages. **C** is incorrect because `com.scjaexam.backing_beans` meets appropriate packaging naming conventions.

Compiling and Interpreting Java Code

10. Which usage represents a valid way of compiling a Java class?

A. `java MainClass.class`

B. `javac MainClass`

C. `javac MainClass.source`

D. `javac MainClass.java`

Answer:

☑ **D.** The compiler is invoked by the `javac` command. When compiling a java class, you must include the filename, which houses the main classes including the `.java` extension.

☒ **A, B,** and **C** are incorrect. **A** is incorrect because `MainClass.class` is bytecode that is already compiled. **B** is incorrect because `MainClass` is missing the `.java` extension. **C** is incorrect because `MainClass.source` is not a valid name for any type of Java file.

11. Which two command-line invocations of the Java interpreter return the version of the interpreter?

A. `java -version`

B. `java --version`

C. `java -version ProgramName`

D. `java ProgramName -version`

Answer:

☑ **A** and **C**. The `-version` flag should be used as the first argument. The application will return the appropriate strings to standard output with the version information and then immediately exit. The second argument is ignored.

☒ **B** and **D** are incorrect. **B** is incorrect because the version flag does not allow double dashes. You may see double dashes for flags in utilities, especially those following the GNU license. However, the double dashes do not apply to the version flag of the Java interpreter. **D** is incorrect because the version flag must be used as the first argument or its functionality will be ignored.

12. Which two command-line usages appropriately identify the class path?

A. `javac -cp /project/classes/ MainClass.java`

B. `javac -sp /project/classes/ MainClass.java`

C. `javac -classpath /project/classes/ MainClass.java`

D. `javac -classpaths /project/classes/ MainClass.java`

Answer:

☑ **A** and **C**. The option flag that is used to specify the classpath is `-cp` or `-classpath`.

☒ **B** and **D** are incorrect because the option flags `-sp` and `-classpaths` are invalid.

13. Which command-line usages appropriately set a system property value?

A. `java -Dcom.scjaexam.propertyValue=003 MainClass`

B. `java -d com.scjaexam.propertyValue=003 MainClass`

C. `java -prop com.scjaexam.propertyValue=003 MainClass`

D. `java -D:com.scjaexam.propertyValue=003 MainClass`

Answer:

☑ **A.** The property setting is used with the interpreter, not the compiler. The property name must be sandwiched between the `-D` flag and the equal sign. The desired value should immediately follow the equal sign.

☒ **B, C,** and **D** are incorrect because `-d`, `-prop`, and "`-D:`" are invalid ways to designate a system property.