SR 9.1 Describe the relationship between a parent class and a child class.

A child class is derived from a parent class using inheritance. The methods and variables of the parent class automatically become a part of the child class, subject to the rules of the visibility modifiers used to declare them.

SR 9.2 How does inheritance support software reuse?

Because a new class can be derived from an existing class, the characteristics of the parent class can be used without the error-prone process of copying and modifying code.

SR 9.3 What relationship should every class derivation represent?

Each inheritance derivation should represent an is-a relationship: the child is a more specific version of the parent. If this relationship does not hold, then inheritance is being used improperly.

SR 9.4 What does the protected modifier accomplish?

The protected modifier establishes a visibility level (like public and private) that takes inheritance into account. A variable or method declared with protected visibility can be referenced by name in the derived class, while retaining some level of encapsulation. Protected visibility allows access from any class in the same package.

SR 9.5 Why is the super reference important to a child class?

The super reference can be used to call the parent’s constructor, which cannot be invoked directly by name. It can also be used to invoke the parent’s version of an overridden method.

SR 9.6 Define a class SchoolBook2 that extends Book2 to include an attribute indicating the age (4 through 16) that a book targets. The constructor accepts the age as a parameter. The class also provides a level method that returns a string as follows: “Pre-school” if the target age is 4 through 6, “Early” if the target age is 7 through 9, “Middle” if the target age is 10 through 12, and “Upper” if the target age is 13 through 16.

public class SchoolBook2 extends Book2 {

private int ageLevel;

SchoolBook2(int totalPages, int age) {

super(totalPages);

ageLevel = age;

}

String getLevel() {

if (ageLevel <= 16 && ageLevel >= 13)

return “Upper”;

if (ageLevel >= 10)

return "Middle”;

if (ageLevel >= 7)

return "Early”;

if (ageLevel >= 4)

return "Pre-school”;

}

}

SR 9.7 What is the difference between single inheritance and multiple inheritance?

With single inheritance, a class is derived from only one parent, whereas with multiple inheritance, a class can be derived from multiple inheritance is that collisions must be resolved in the cases when two or more parents contribute an attribute or method with the same name. Java supports only single inheritance.

SR 9.8 Why would a child class override one or more of the methods of its parent class?

A child class may prefer its own definition of a method in favor of the definition provided for it by its parent. In this case, the child overrides (redefines) the parent’s definition with its own.

SR 9.9 True or False? Explain.

a. A child class may define a method with the same name as a method in the parent class.

True. The child “overrides” the parent’s definition of the method if both methods have the same signature.

b. A child class can override the constructor of the parent class.

False. A constructor is a special method with no return type which has the same name as its class. If you tried to override the parent’s constructor, you would create a syntax error since all methods except constructors must have a return type.

c. A child class can override a final method of the parent class.

False. A final method cannot be overridden (that’s why it is “final”).

d. It is considered poor design when a child class overrides a method from the parent class.

False. On the contrary, the need to override methods of a parent class occurs often when using inheritance.

e. A child class may define a variable with the same name as a variable in the parent class.

True. Such a variable is called a shadow variable. You can do this, but it may lead to confusing situations and its use is discouraged.

SR 9.10 Draw a UML class diagram showing an inheritance hierarchy containing classes that represent different types of food. Show some appropriate variables and method names for at least two of these classes.

SR 9.11 What is the significance of the Object class?

All classes in Java are derived, directly or indirectly, from the Object class. Therefore, all public methods of the Object class, such as equals and toString, are available to every object.

SR 9.12 Which is the only Java class that does not have a parent class? Explain.

The only Java class that does not have a parent class is the Object class. As mentioned in the previous answer, all other classes are derived from Object either directly or indirectly. Object is the root of the Java inheritance tree.

SR 9.13 What is the role of an abstract class?

An abstract class is a representation of a general concept. Common characteristics and method signatures can be defined in an abstract class so that they are inherited by child classes derived from it.

SR 9.14 Why is it a contradiction to define a final, abstract class?

It is a contradiction to an abstract class as final. An abstract class cannot be instantiated and often contains abstract methods. Typically, the definitions of these methods are completed by one or more classes that extend the abstract class. But a final class cannot be extended, so there would be no way to complete its definition.

SR 9.15 What is an interface hierarchy?

A new interface can be derived from an existing interface using inheritance, just as a new class can derived from an existing class.

SR 9.16 Are all members of a parent class inherited by the child? Explain.

A class member is not inherited if it has private visibility, meaning that it cannot be referenced by name in the child class.

SR 9.17 Could the Pizza class refer to the variable servings explicitly? What about the calories method? Explain.

The Pizza class can refer to the variable servings explicitly because it is declared with protected visibility. It cannot, however, refer to the calories method explicitly, because the calories method is declared as private.

SR 9.18 What does it mean for an inheritance derivation to represent an is-a relationship?

An inheritance derivation represents an “is-a” relationship when the child class represents a more specific version of the parent class. For example, a dictionary is a type of book, so if a Dictionary class extends a Book class, the inheritance represents an “is-a” relationship.

SR 9.19 Where should common features of classes appear in a class hierarchy? Why?

Common features of class should appear as high as possible in a class hierarchy, as long as it is appropriate for the features to be at the level where they are defined. This approach supports understandability, consistency, and reuse.

SR 9.20 How can you define a class with multiple roles?

SR 9.21 Why should you override the toString method of a parent in its child class, even when the method is not invoked through the child by your current applications?

SR 9.22 How can the final modifier be used to restrict inheritance? Why would you do this?

SR 9.23 What is the purpose of the Shape class?

SR 9.24 Describe the inheritance relationship between a Node and an Ellipse.

SR 9.25 Describe the inheritance relationship between a Node and a Label.

SR 9.26 Which nodes can serve as root nodes of a Scene?

SR 9.27 What are some ways a color picker allows the user to specify a color?

SR 9.28 What does the getValue method of a DatePicker object return? What does the getValue method of a ColorPicker object return?

SR 9.29 What is a dialog box?

SR 9.30 What classes can be used to create and display dialog boxes?

SR 9.31 What is a file chooser?

SR 10.1 What is polymorphism?

SR 10.2 Why is compile time binding considered more efficient than dynamic binding?

SR 10.3 How does inheritance support polymorphism?

SR 10.4 Suppose the class MusicPlayer is the parent of the class CDPlayer. Is the following sequence of statements allowed in Java? Explain.

MusicPlayer mplayer = new MusicPlayer();

CDPlayer cdplayer = new CDPlayer();

mplayer = cdplayer;

SR 10.5 Suppose the class MusicPlayer is the parent of the class CDPlayer. Is the following sequence of statements allowed in Java? Explain.

MusicPlayer mplayer = new MusicPlayer();

CDPlayer cdplayer = new CDPlayer();

cdplayer = mplayer;

SR 10.6 How is overriding related to polymorphism?

SR 10.7 Why is the StaffMember class in the Firm example declared as abstract?

SR 10.8 Why is the pay method declared in the StaffMember class, given that it is abstract and has no body at that level?

SR 10.9 Which pay method is invoked by the following line from the payday method of the Staff class? amount = staffList[count].pay();

SR 10.10 How can polymorphism be accomplished using interfaces?

SR 10.11 Suppose that the Speaker interface and the Philosopher and Dog classes are as described in this section. Are the following sequences of statements allowed in Java? Explain.

a.

Speaker current = new Speaker();

b.

Speaker current = new Dog();

c.

Speaker first, second;

first = new Dog();

second = new Philosopher();

first.speak();

first = second;

d.

Speaker first = new Dog();

Philospher second = new Philosopher();

second.pontificate();

first = second;

e.

Speaker first = new Dog();

Philospher second = new Philosopher();

first = second;

second.pontificate();

first.pontificate();

SR 10.12 Describe the Comparable<T> interface.

SR 10.13 Show the sequence of changes the selection sort algorithm makes to the following list of numbers:

5 7 1 8 2 4 3

SR 10.14 Show the sequence of changes the insertion sort algorithm makes to the following list of numbers:

5 7 1 8 2 4 3

SR 10.15 In what way are the sort methods defined in this chapter polymorphic?

SR 10.16 Which is better: selection sort or insertion sort? Explain.

SR 10.17 Given the following list of numbers, how many elements of the list would be examined by the linear search algorithm to determine if each of the indicated target elements are on the list?

15 21 4 17 8 27 1 22 43 57 25 7 53 12 16

a. 17

b. 15

c. 16

d. 45

SR 10.18 Describe the general concept of a binary search.

SR 10.19 Given the following list of numbers, how many elements of the list would be examined by the binary search algorithm to determine if each of the indicated target elements are on the list?

1 4 7 8 12 15 16 17 21 22 25 27 43 53 57

a. 17

b. 15

c. 57

d. 45

SR 10.20 Suppose you are designing classes for a banking-related system. Both checking accounts and savings accounts require deposit and withdraw operations. You decide to provide these behaviors using polymorphism. Which polymorphic mechanism (inheritance or interfaces) is best suited for this situation? Provide support for your choice.

SR 10.21 Suppose you are designing classes to help create aquarium-based screen savers. At times, you will want some aquarium objects to “float” from wherever they are to the top of the tank. You decide to provide this behavior using polymorphism. Which polymorphic mechanism (inheritance or interfaces) is best suited for this situation? Provide support for your choice.

SR 10.22 Suppose you are designing classes to support the modeling of rain forest environments. Animal objects, such as butterflies and monkeys, need to grow older periodically. You decide to provide this behavior using polymorphism. Which polymorphic mechanism (inheritance or interfaces) is best suited for this situation? Provide support for your choice.

SR 10.23 What is a JavaFX property?

SR 10.24 What is the result of property binding?

SR 10.25 How do you perform mathematical operations on numeric properties?

SR 10.26 What is a change listener?

SR 10.27 What does a slider allow the user to do?

SR 10.28 How do you access the value that a slider represents?

SR 10.29 How does the user interact with a spinner?

SR 10.30 Why might a program designer opt to use a spinner rather than a choice box?

SR 10.31 What is a spinner value factory?

SR 11.1 What is the difference between an error and an exception?

SR 11.2 In what ways might a thrown exception be handled?

SR 11.3 True or False. Explain.

a. An exception and an error are the same thing.

b. An attempt to divide by zero will cause an exception to be thrown.

c. If a program does not handle a raised exception, the exception is ignored and nothing happens.

d. If a program does not handle an exception, a message related to the exception will be produced.

e. A call stack trace shows the sequence of method calls that led to the code where an exception occurred.

SR 11.4 What is a catch clause?

SR 11.5 What is a finally clause?

SR 11.6 What output is produced by the following code fragment under each of the stated conditions?

try

{

review.question();

}

catch (Exception1 exception)

{

System.out.println("one caught");

}

catch (Exception2 exception)

{

System.out.println("two caught");

}

finally

{

System.out.println("finally");

}

System.out.println("the end");

a. No exception is thrown by the review.question() method.

b. An Exception1 exception is thrown by the review.question() method.

c. An Exception2 exception is thrown by the review.question() method.

d. An Exception3 exception is thrown by the review.question() method.

SR 11.7 What happens if an exception is not caught?

SR 11.8 How would the result of the Propagation program change if the following code fragment was placed in the level2 method just before the call to the level3 method?

int num = 10, den = 0;

int res = num / den;

SR 11.9 How would the result of the Propagation program change if the following code fragment was placed in the level2 method just after the call to the level3 method?

int num = 10, den = 0;

int res = num / den;

SR 11.10 What is a checked exception?

SR 11.11 True or False? Explain.

a. An ArithmeticException is an Exception.

b. An ArithmeticException is Throwable.

c. An ArithmeticException is a checked exception.

d. A NoSuchMethodException is a checked exception.

e. We can create our own exceptions by extending the Exception class.

f. A throws clause must be appended to the header of a method definition if the method may potentially throw an ArithmeticException.

SR 11.12 What happens if the input to the CreatingExceptions program is 42? What if it is -3?

SR 11.13 What is a stream?

SR 11.14 What are the standard I/O streams?

SR 11.15 What Stream class object have we been using explicitly throughout this book?

SR 11.16 An I/O exception, the InputMismatchException, will occur during the main method of the CreatingExceptions program (see Listing 11.5) if the user enters an alphabetic character. Why doesn’t the main method definition include a throws InputMismatchException clause?

SR 11.17 An I/O exception, the FileNotFoundException, will occur during the main method of the TestData program if the test.txt is not writable. Why doesn’t the main method definition include a throws FileNotFoundException clause?

SR 11.18 What is the purpose of the close method of the PrintWriter class?

SR 11.19 What is a tool tip?

SR 11.20 Why might you want to disable a control?

SR 11.21 What does a scroll pane do?

SR 11.22 What are the three ways the user can change the knob position on a scroll bar?

SR 11.23 What does the scroll bar policy determine?

SR 11.24 Describe the role of the divider bar on a split pane.

SR 11.25 What is the difference between a choice box and a list view?

SR 11.26 What happens if more than two nodes are added to a split pane?

SR 12.1 What is recursion?

SR 12.2 How many times is the recursive part of the definition of a List used to define a list of 10 numbers? How many times is the base case used?

SR 12.3 What is infinite recursion?

SR 12.4 When is a base case needed for recursive processing?

SR 12.5 Write a recursive definition of 5 \* n (integer multiplication), where n > 0. Define the multiplication process in terms of integer addition. For example, 5 \* 7 is equal to 5 added to itself 7 times.

SR 12.6 Is recursion necessary?

SR 12.7 When should recursion be avoided?

SR 12.8 Describe what is returned by the following recursive method.

public int exercise(int n)

{

if (n < 0)

return –1;

else

if (n < 10)

return 1;

else

return 1 + exercise(n/10);

}

SR 12.9 Write a recursive method that returns the value of 5 \* n, where n > 0. See Self-Review Question 12.5 . Explain why you would not normally use recursion to solve this problem.

SR 12.10 What is indirect recursion?

SR 12.11 Under what conditions does the recursion stop in the

MazeSearch program?

SR 12.12 Identify where in the MazeSearch program each of

the following is provided.

a. The original maze is defined.

b. A test to see if we have arrived at the goal occurs.

c. A location is marked as having been tried.

d. A test to see if we already tried a location occurs.

SR 12.13 Trace the MazeSearch program to determine the

series of calls to the method valid (including the values of the

parameters that are passed) that would occur if the original

maze is as shown.

a. 1 1

1 1

b. 0 0

0 0

c. 1 1

1 0

SR 12.14 Explain the general approach to solving the Towers of Hanoi puzzle. How does it relate to recursion?

SR 12.15 Trace the SolveTowers code for an initial stack of 1 disk. How many calls to the moveTower method are made? How many calls are made for an initial stack of 2 disks? How many for 3 disks? Describe a pattern related to the number of calls made to the moveTower method as the number of disks increases.

SR 12.16 Where does recursion occur in the TiledImages program? What does it accomplish?

SR 12.17 How many Image objects are created in the TiledImages program? How many ImageView objects?

SR 12.18 What is the base case of the recursion in the TiledImages program?

SR 12.19 What is a fractal? What does it have to do with recursion?

SR 12.20 Why does the program impose an upper limit on the order of the Koch snowflake fractal?

SR 12.21 Describe how each line segment of a Koch snowflake is changed when going to the next higher order fractal.

SR 13.1 What is a collection?

SR 13.2 What’s the difference between a collection and a data structure?

SR 13.3 Why are objects particularly well suited for implementing abstract data types?

SR 13.4 What is a dynamic data structure?

SR 13.5 Describe the steps depicted in Figure 13.2 to insert a node into a list. What special cases exist?

SR 13.6 Describe the steps depicted in Figure 13.3 to delete a node from a list. What special cases exist?

SR 13.7 Suppose first is a reference to a Node object, and that it refers to the first node in a linked list. Show, in pseudocode, the steps that would count and return the number of nodes on the list.

SR 13.8 What is a doubly linked list?

SR 13.9 What is a header node for a linked list?

SR 13.10 How is a queue different from a list?

SR 13.11 Show the contents of a queue after the following operations are performed. Assume the queue is initially empty.

enqueue(5);

enqueue(21);

dequeue();

enqueue(72);

enqueue(37);

enqueue(15);

dequeue();

SR 13.12 What is a stack?

SR 13.13 Show the contents of a stack after the following operations are performed. Assume the stack is initially empty.

push(5);

push(21);

pop();

push(72);

push(37);

push(15);

pop();

SR 13.14 What is the Stack class?

SR 13.15 What do trees and graphs have in common?

SR 13.16 Which structure (a tree or a graph) would be a good choice to represent each of the following.

a. The directories and files on a computer system.

b. Airplane routes.

c. An “is a friend of” relationship among a group of people.

d. An “is a boss of” relationship in a company.

SR 13.17 What is the Java Collections API?

SR 13.18 What is a generic type, and how does it relate to the Java Collections API?