PISCATAWAY
TOWNSHIP SCHOOLS

Dr. Frank Ranelli
Superintendent of Schools

Dr. William Baskerville
Assistant Superintendent

Java Certification & Artificial Intelligence with Machine Learning

Content Area: **Mathematics**

Grade Span: **9–12**

Revised by: **Jackie Puleio**

Title **Supervisor of Mathematics, 7–12**

Approval Date:

Members of the Board of Education

Piscataway Township Schools
1515 Stelton Road
Piscataway, NJ 08854–1332
732 572–2289, ext. 2561
Fax 732 572–1540
www.piscatawayschools.org

COURSE OVERVIEW

Description

Java Certification and Artificial intelligence with Machine Learning (JCAIML), focuses on developing a dual semester advanced computer science course combining Java Fundamentals, Programming and Machine Learning. The goals of instituting this course into the high school curriculum is to enable students to obtain their Junior Oracle Certified Associate (OCA Jr.) and Oracle Certified Associate (OCA) certifications. This will allow students to become a viable candidate for internships and to encourage students to pursue post–secondary career in computer science. This course will ultimately prepare students to obtain certifications in Java, providing experiences for any student enrolled to obtain an experience that parallels experiences in local colleges and/or technical institutions.

Goals

In addition to the content standards, skills, and concepts set forth, this course also seeks to meet the Standards for Mathematical Practice set forth in the New Jersey Student Learning Standards. These practices include generally applied best practices for learning mathematics, such as understanding the nature of proof and having a productive disposition towards the subject, and are not tied to a particular set of content.

The eight Standards for Mathematical Practice are outlined below:

1. Make sense of problems and persevere in solving them.
2. Reason abstractly and quantitatively.
3. Construct viable arguments and critique the reasoning of others.
4. Model with mathematics.
5. Use appropriate tools strategically.
6. Attend to precision.
7. Look for and make use of structure.
8. Look for and express regularity in repeated reasoning.

The goals of instituting this course into the high school curriculum will enable students to obtain their Oracle Certified Junior Associate (Java Foundations) and Oracle Certified Associate Certification Exam (Java SE 8 Programmer I) to further encourage pursuit of a post–secondary career in computer science.

Scope and Sequence

Unit	Topic	Length
Unit 1	Java Fundamentals	7 days
Unit 2	Working with Java data types	8 days
Unit 3	Methods and Encapsulation	15 days
Unit 4	Selected classes from the Java API and Arrays	15 days
Unit 5	Flow Control	15 days
Unit 6	Working with Inheritance	20 days
Unit 7	Exception Handling	20 days

Resources

Core Text:

OCA Java SE 8 Programmer I Certification Guide (2016). Gupta, Mala (ISBN# 9781617293252)
Oracle Certified Associate Java SE 8 Programmer I (ISBN# ISBN# 9781118957400)

Suggested Resources:

https://education.oracle.com/java–foundations/pexam_1Z0–811
https://education.oracle.com/java–se–8–programmer–i/pexam_1Z0–808

UNIT 1: Java Fundamentals

Summary and Rationale

Java classes are structured with several components that form a hierarchy. These include modifiers that both allow and disallow access to classes, and packages that can be imported into classes to add features. This unit imparts the essential knowledge of Java classes' structure, including: the differences between a class and Java source code, understanding the structure of a Java class (including the "main" method), understanding how to import packages, understanding abstract classes, understanding the "static" keyword, and applications of access modifiers. In addition, students learn which features/components of Java (encapsulation, platform independence, object orientation) are relevant to know.

Recommended Pacing

7 days

Instructional Focus

Unit Enduring Understandings

This unit covers the fundamentals of Java. Students will see how to define and run a Java class, and learn about packages, variables, and the object life cycle.

Unit Essential Questions

- What is the structure of a Java class and source code file?
- What are executable Java applications?
- How are Java packages defined?
- What is the purpose of the packages in java?
- How can you use a class from a package?
- Can we import all of the classes of a package with one statement?
- Can we import an individual static member of a class?
- What is the purpose of access modifiers?
- What is the relationship between all of the four access levels and all the threeaccess modifiers?
- What are non access modifiers?
- How are abstract classes different from interfaces?
- Can we define variable as abstract?
- What is the purpose of the static modifier in a class and how can we use it?
- What are the major components and features of the Java Language?

Objectives

Students will know:

- Structure of a Java class, with its components: package and import statements, class declarations, comments, variables, and methods
- Difference between the components of a Java class and that of a Java source code file
- The right method signature for the main method to create an executable Java application
- The arguments that are passed to the main method
- Understand packages and import statements
- Get the right syntax and semantics to import classes from packages and interfaces in your own classes
- Application of access modifiers (public, protected, default, and private) to a class and its members. Determine the accessibility of code with these modifiers
- The implication of defining classes, interfaces, and methods as abstract entities
- The implication of defining fields and methods as static members
- The features and components that are relevant or irrelevant to Java

Students will be able to:

- [1.2] Define the structure of a Java class
- [1.3] Create executable Java applications with a main method; run a Java program from the command line; including console output
- [1.4] Import other Java packages to make them accessible in your code
- [6.4] Apply access modifiers
- [7.5] Use abstract classes and interfaces
- [6.2] Apply the static keyword to methods and fields
- [1.5] Compare and contrast the features and components of Java such as: platform independence, object orientation, encapsulation, etc.

Resources

Core Text:

OCA Java SE 8 Programmer I Certification Guide (2016). Gupta, Mala (ISBN# 9781617293252)
Oracle Certified Associate Java SE 8 Programmer I (ISBN# ISBN# 9781118957400)

Suggested Resources:

https://education.oracle.com/java–foundations/pexam_1Z0–811
https://education.oracle.com/java–se–8–programmer–i/pexam_1Z0–808

UNIT 2: Working with Java data types

Summary and Rationale

Java stores its data with multiple data types, and they are stored in variables. Java uses different types of operators to manipulate the variables and data. This unit compares and contrasts the data types and variables, and how to create and initialize them. In addition, students learn how to use the fourtypes of operators with the variables, and when students can/cannot use operators with certain variables. Lastly, this unit covers wrapper classes, their position in the hierarchy, their objects and values, and autoboxing/unboxing.

Recommended Pacing

8 days

Instructional Focus

Unit Enduring Understandings

Like many programming languages, Java is composed primarily of variables, operators, and statements put together in some logical order. In the previous unit, variables were discussed and some examples were considered; in this unit we'll discuss the various operators and statements available to students within the language. This knowledge will allow students to build complex functions and class structures that they will use in later units.

Unit Essential Questions

- What are primitive variables? Define and list examples
- What are the similarities and differences between primitive data types?
- What are some examples/scenarios of when certain primitive data types should/shouldn't/can't be used?
- What are object reference variables?
- What is the relationship between primitive variables and object reference variables?
- What are the four types of integer literal values for primitive variables?
- How do you declare and initialize primitive variables? Object reference variables?
- What conditions create a valid identifier?
- What is the function of a wrapper class?
- How do you create objects of a wrapper class?
- What happens when you add a primitive value to a wrapper class variable?
- What does immutable mean? Are wrapper classes immutable?
- What are the functions of the four operators?
- How do you use operators with both primitive and object reference variables?
- How do you determine if two primitives are equal?
- How do you use parentheses to modify/override default operator precedence?

Objectives

Students will know:

- The primitive data types in Java, including scenarios when a particular primitive data type should or can't be used
- Similarities and differences between the primitive data types
- Similarities and differences between primitive and object reference variables
- Declaration and initialization of primitives and object reference variables
- Literal values for primitive and object reference variables
- How and when values are boxed and unboxed when used with wrapper classes
- Use of assignment, arithmetic, relational, and logical operators with primitives and object reference variables
- Valid operands for an operator

- Output of an arithmetic expression
- Determine the equality of two primitives
- How to override the default operator precedence by using parentheses

Students will be able to:

- [2.2] Differentiate between object reference variables and primitive variables
- [2.1] Declare and initialize variables (including casting of primitive data types)
- [2.5] Develop code that uses wrapper classes such as Boolean, Double, and Integer
- [3.1] Use Java operators; including parenthe– ses to override operator precedence

Resources

Core Text:

OCA Java SE 8 Programmer I Certification Guide (2016). Gupta, Mala (ISBN# 9781617293252)
Oracle Certified Associate Java SE 8 Programmer I (ISBN# ISBN# 9781118957400)

Suggested Resources:

https://education.oracle.com/java–foundations/pexam_1Z0–811
https://education.oracle.com/java–se–8–programmer–i/pexam_1Z0–808

UNIT 3: Methods and Encapsulation

Summary and Rationale

Making sure an object is well–encapsulated is a crucial part of creating a class that doesn't expose the object. Using a set of methods that interact with the person using a class, and the internal, encapsulated object, students can ensure their object is well–encapsulated. This protects it from being assigned undesirable values and becoming unstable. In this unit, first, variables' scopes are explored, along with the scopes of method parameters. The object's life cycle is explained: it describes the period when an object has been initialized and is being referenced, and has not gone out of scope. This unit also covers methods, how they return values, and how to overload them. Constructors of a class and overloaded constructors are included, along with the access levels that can be used to define them. The differences between initializers and constructors is reviewed, and the "object field" is introduced. Lastly, the unit describes the differences in behavior when an object and when a primitive are passed to a method.

Recommended Pacing

15 days

Instructional Focus

Unit Enduring Understandings

In the previous units, students have used methods and constructors without examining them in detail. In this unit, students will explore methods and constructors in depth and cover everything they will need to know about them for the OCA exam. This unit discusses instance variables, the final keyword, access modifiers, and initialization. Students will also learn how to write a simple lambda expression.

Unit Essential Questions

- What is/are the scope(s) of a variable?
- What is the difference between local and loop variables?
- Limitations of local variables(accessing, defining)
- What are class variables?
- Method parameters and their scope.
- What is "object's life cycle"? What is an "alive" object?
- Describe how a method returns values and how you know what kind of value it will return
- What is an overloaded method? How do you create one?
- What are constructors? Do they return anything?
- What four access levels can you use to define a constructor? What does “default” accessibility mean?
- What is an initializer block? How does this relate to a constructor? Compare/Contrast.
- What is an overloaded constructor? How can a constructor call another overloaded constructor?
- What is an object field? How can it be read? How can it be written/manipulated?
- What is the difference between a well–encapsulated object and one that isn't?
- Contrast what happens when an object is passed to a method and when a primitive is passed to a method.

Objectives

Students will know:

- Variables can have multiple scopes: class, instance, method, and local
- Accessibility of a variable in a given scope
- Object fields can be read from and written to by directly accessing instance variables and calling methods
- The correct notation to call methods on an object. Methods may or may not change the value of instance variables
- Access modifiers affect access to instance variables and methods that can be called using a reference variable

- Nonstatic methods can't be called on uninitialized objects
- Differences between when an object is declared, initialized, accessible, and eligible to be collected by Java's garbage collection
- Garbage collection in Java
- Creation of methods with correct return types and method argument lists
- Creation of methods with the same names, but a different set of argument lists

Students will be able to:

- [1.1] Define the scope of variables
- [2.3] Know how to read or write to object fields
- [2.4] Explain an Object's Lifecycle (creation, "dereference by reassignment" and garbage collection)
- [6.1] Create methods with arguments and return values; including overloaded methods

Resources

Core Text:

OCA Java SE 8 Programmer I Certification Guide (2016). Gupta, Mala (ISBN# 9781617293252)
Oracle Certified Associate Java SE 8 Programmer I (ISBN# ISBN# 9781118957400)

Suggested Resources:

https://education.oracle.com/java–foundations/pexam_1Z0–811
https://education.oracle.com/java–se–8–programmer–i/pexam_1Z0–808

UNIT 4: Selected classes from the Java API and Arrays

Summary and Rationale

This unit covers String objects, StringBuilder objects, Arrays, ArrayLists, and date/time objects that will be present in the certification exam. The methods for initializing and manipulating Strings are reviewed, along with utilizing methods from String and StringBuilder. The unit describes the different types of arrays and how to create them, and how to assess their elements. ArrayLists' usage and advantages over Arrays are covered, along with using methods for adding, modifying, deleting, and manipulating an Arraylist's elements. The unit also describes the usage of LocalDate, LocalTime, and LocalDateTime classes. Usage of the Period class with object manipulation, and the usage of DateTimeFormatter are also explained.

Recommended Pacing

15 days

Instructional Focus

Unit Enduring Understandings

The OCA exam expects students to know the core data structures and classes used in Java, and in this unit students will learn about the most common methods available. For example, String and StringBuilder are used for text data. An array and an ArrayList are used when code has multiple values. A variety of classes are used for working with dates. In this unit, students will also learn how to determine whether two objects are equal.

API stands for application programming interface. In Java, an interface is something special. In the context of an API, it can be a group of class or interface definitions that gives students access to a service or functionality. Students will learn about the most common APIs for each of the classes covered in this unit.

Unit Essential Questions

- What is the String class?
- How do you initialize a String so it is placed in a pool? When is it not placed in a pool?
- What is the difference between using == and "equals" for comparing Strings?
- How you can manipulate/search through strings(using "charAt", "trim", "endsWith", etc.)?
- What is the StringBuilder class? How is it different from the String class?
- What is an array? What types of arrays exist(dimension–wise)?
- What is the significance of all arrays being Objects?
- What is an ArrayList? How does it compare to an array?
- What are some ways an ArrayList and its contents can be manipulated?
- How does the "equals" method work? What does it compare?
- What are LocalDate and LocalTime? What format do they use?
- How can LocalDate and LocalTime be manipulated?
- What is LocalDateTime?
- What is the Period class? How is it used? What interface does it implement?
- What is the functionality and usage of DateTimeFormatter?
- What is an asymmetrical multidimensional Array? How do you access its elements?

Objectives

Students will know:

- How to initialize String variables using = (assignment) and new operators. Use of the operators =, +=, !=, and == with String objects
- Pooling of string literal values. Literal value for class String
- Use of methods from class String

- Immutable String values
- All the String methods manipulate and return a new String object
- How to determine the equality of two String objects
- Differences between using operator == and method equals() to determine equality of String objects
- How to create StringBuilder classes and how to use their commonly used methods
- Difference between StringBuilder and String classes
- Difference between methods with similar names defined in both of these classes
- How to declare, instantiate, and initialize one–dimensional arrays using single and multiple steps
    - The dos and don'ts of each of these steps
- How to declare, instantiate, and initialize multi–dimensional arrays using single and multiple steps, with the dos and don'ts for each of these steps
- Accessing elements in asymmetric multidimensional arrays

Students will be able to:

- [9.2] Creating and manipulating Strings
- [3.2] Test equality between Strings and other objects using == and equals()
- [9.1] Manipulate data using the StringBuilder class and its methods
- [4.1] Declare, instantiate, initialize, and use a one–dimensional array
- [4.2] Declare, instantiate, initialize, and use a multidimensional array

Resources

Core Text:

OCA Java SE 8 Programmer I Certification Guide (2016). Gupta, Mala (ISBN# 9781617293252)
Oracle Certified Associate Java SE 8 Programmer I (ISBN# ISBN# 9781118957400)

Suggested Resources:

https://education.oracle.com/java–foundations/pexam_1Z0–811
https://education.oracle.com/java–se–8–programmer–i/pexam_1Z0–808

Unit 5: Flow Control

Summary and Rationale

Using "if", "if–else", ternary, and "switch", constructs to execute something only under certain conditions, along with using loops to continue an action, are essential parts of writing any code. This unit covers the syntax of "for"enhanced "for", "do–while", and "while" loops along with when they should and shouldn't be used. In addition, the conditional statements are explained and contrasted with each other. Nested if–else statements are also covered in this unit. The advantages and disadvantages of using loops and using conditionals are compared for different situations. The usage of "break" and "continue" statements, and their uses with labeled statements and loops, are also described.

Recommended Pacing

15 days

Instructional Focus

Unit Enduring Understandings

In real life, we all make multiple decisions on a daily basis, and we often have to choose from a number of available options to make each decision. These decisions range from the complex, such as selecting what subjects to study in school or which profession to choose, to the simple, such as what food to eat or what clothes to wear. We may also repeat particular sets of actions. These actions can range from eating ice cream cone every day, to calling a friend until you connect, or to passing exams at school in order to achieve a desired degree.

In Java, the selection statements (if and switch) and looping statements (for, enhanced for, while, and do-while) are used to define and choose among different courses of action, as well as to repeat the lines of code. We use these types of statements to define the flow of control in code.

In the OCA Java SE 8 Programmer I exam, students will be asked how to define and control the flow in your code.

Unit Essential Questions

- What is an "if" statement? What are some different types? Where is the "then" part implied to be?
- What is the syntax for nested if–else–if–else statements?
- What do if statements usually return?
- What is a ternary operator and its syntax for usage? When is it used?
- What is a switch statement used for? What statement is used to exit a switch construct?
- What arguments does a switch statement accept?
- What is a case value? What are its limitations?(for example, it cannot be null)
- What is a for loop and when is it traditionally used?
- What is the syntax for regular and nested for loops?
- What is the relationship between an enhanced for loop/for–each loop and a regular loop?
- In what situation is it advantageous to use a regular for loop? An enhanced for loop?
- What is the function of a while loop? What is the function of a do–while loop?
- What is the relationship between while and do–while loops. In what situation is it advantageous to use each?
- What is the relationship between while, do–while, for, and enhanced–for loops?
- For each type of loop, what are situations where it is advantageous to use that loop over the other three?
- What is the break statement used for? What happens when you use it with a nested loop?
- What is the continue statement used for? What happens when you use it with a nested loop?
- Labeled statements: What can you add labels to? What can't you add labels to?
- How can you use labeled break and labeled continue statements?

Objectives

Students will know:

- How to use if, if–else, if–else–if–else, and nested if constructs
- The differences between using these if constructs with and without curly braces {}
- How to use a ternary construct. How it compares with if and if–else constructs
- How to use a switch statement by passing the correct type of arguments to the switch statement and case and default labels
- The change in the code flow when break and return statements are used in the switch statement
- How to use the while loop, including determining when to apply the while loop
- How to use for and enhanced for loops. The advan– what advantages and disadvantages of the for loop and enhanced for loop. Scenarios when you may not be able to use the enhanced for loop
- Creation and use of do–while loops. Every do–while loop executes at least once, even if its condition evalu–ates to false for the first iteration

Students will be able to:

- [3.3] Create if and if/else and ternary constructs
- [3.4] Use a switch statement
- [5.1] Create and use while loops
- [5.2] Create and use for loops including the enhanced for loop
- [5.3] Create and use do/while loops

Resources

Core Text:

OCA Java SE 8 Programmer I Certification Guide (2016). Gupta, Mala (ISBN# 9781617293252)
Oracle Certified Associate Java SE 8 Programmer I (ISBN# ISBN# 9781118957400)

Suggested Resources:

https://education.oracle.com/java–foundations/pexam_1Z0–811
https://education.oracle.com/java–se–8–programmer–i/pexam_1Z0–808

Unit 6: Working with Inheritance

Summary and Rationale

Java classes can inherit variables and methods from their "parent" class, similar to how living things inherit traits from parents. Different additional methods and variables can be defined by classes, to distinguish them from their parent class. Inheritance has several benefits which are crucial to creating efficient classes. This unit covers how to implement inheritance from a parent class to another class. In addition, the unit goes over polymorphism and how to implement it with classes and interfaces, including abstract ones. Casting, and when it should be used, along with how to cast an object to a class/interface, are explained. This unit also covers using "super" and "this" to access variables, methods, and constructors. Lastly, the use of the Predicate class and the syntax and usage of lambda expressions are explained.

Recommended Pacing

20 days

Instructional Focus

Unit Enduring Understandings

In this unit, we will take things one step further and show how class structure is one of the most powerful features in the Java language. At its core, proper Java class design is about code reusability, increased functionality, and standardization. For example, by creating a new class that extends an existing class, students may gain access to a slew of inherited primitives, objects, and methods. Alternatively, by designing a standard interface for their application, students ensure that any class that implements the interface have certain required methods defined. Finally, by creating abstract class definitions, students will define a platform that other developers can extend and elaborate.

Unit Essential Questions

- What are the benefits of inheritance? What can a class inherit? What can interfaces inherit?
- What is a derived class?
- What are the functions of the keywords "extends" and "implements"?
- How does inheritance work with abstract and concrete classes?
- What rules have to be followed for a class to extend multiple interfaces?
- Is multiple inheritance allowed in Java? Why/why not?
- How can you use inheritance to refer to an object of a derived class?
- What is casting? How is it implemented? Why is it needed?
- What are the functions of "super" and "this"? How are they different?
- When does polymorphism come into play in Java?
- How should overridden methods be implemented? (name, argument, method parameters, access)
- If a method is defined in the base class and overloaded in the derived class, is it polymorphic?
- Do static methods in interfaces participate in polymorphism?
- What sort of interfaces do lambdas work with?
- For each lambda expression, which sections(i.e. lambda body, arrow) are mandatory or optional?

Objectives

Students will know:

- The need for inheriting classes
- How to implement inheritance using classes
- How to implement polymorphism with classes and interfaces
- How to define polymorphic or overridden methods
- How to determine the valid types of the variables that can be used to refer to an object

- How to determine the differences in the members of an object, which ones are accessible, and when an object is referred to using a variable of an inherited base class or an implemented interface
- The need for casting
- How to cast an object to another class or an interface
- How to access variables, methods, and constructors using super and this
- What happens if a derived class tries to access variables of a base class when the variables aren't accessible to the derived class
- The role of abstract classes and interfaces in implementing polymorphism
- Syntax and usage of lambda expressions
- Usage of Predicate class

Students will be able to:

- [7.1] Describe inheritance and its benefits
- [7.2] Develop code that demonstrates the use of polymorphism; including overriding and object type versus reference type
- [7.3] Determine when casting is necessary
- [7.4] Use super and this to access objects and constructors
- [7.5] Use abstract classes and interfaces
- [9.5] Write a simple Lambda expression that consumes a Lambda Predicate expression

Resources

Core Text:

OCA Java SE 8 Programmer I Certification Guide (2016). Gupta, Mala (ISBN# 9781617293252)
Oracle Certified Associate Java SE 8 Programmer I (ISBN# ISBN# 9781118957400)

Suggested Resources:

https://education.oracle.com/java–foundations/pexam_1Z0–811
https://education.oracle.com/java–se–8–programmer–i/pexam_1Z0–808

Unit 7: Exception Handling

Summary and Rationale

Exception handlers are necessary to keep code flowing smoothly, in the case of an exceptional condition. Exceptional conditions can modify how the code works, but an exception handler will ensure the code works correctly if an exceptional condition occurs. This unit covers how to identify exceptions and common exception categories/classes in code. The unit explains how to understand exceptions and how they happen, along with how they affect the normal code flow. In addition, the importance of separate exception handlers in students' code is emphasized. The unit goes over using "try–catch–finally" blocks to handle exceptions, and the differences between checked exceptions, unchecked exceptions, and errors. Lastly, the unit goes over how to invoke methods that throw exceptions.

Recommended Pacing

20 days

Instructional Focus

Unit Enduring Understandings

Many things can go wrong in a program. Java uses exceptions to deal with some of these scenarios. The OCA exam covers the basics of working with exceptions

Unit Essential Questions

- Why should exceptions be handled separately?
- What are the categories common exceptions can be divided into?
- Define checked exception, runtime exception, and error.
- How do you create a method that throws an exception?
- How is the "throws" clause used?
- What happens internally when an exception is thrown? What class is an exception an object of?
- How would you create a nested exception handler?
- List some commonly–occurring exceptions(i.e. NullPointerException)
- What three things can cause an ExceptionInInitializer Error?
- What causes the StackOverflowError?
- What must you do in order to use a method that throws a checked exception? (three options)
- Compare/contrast the effects of a runtime exception, a checked exception, and an error, on a program.
- How do try–catch–finally blocks work with exceptions?
- What are some common mathematical exceptions? What causes them?

Objectives

Students will know:

- The need for and advantages of exception handlers.
- Differences and similarities between checked exceptions, RuntimeExceptions, and Errors. Differences and similarities in the way these exceptions and errors are handled in code.
- How to create a try–catch–finally block. Under– stand the flow of code when the enclosed code throws an exception or error.
- How to create nested try–catch–finally blocks.
- How to create methods that throw exceptions.
- Rules that cover when overriding or overridden methods throw or don't throw exceptions.
- How to determine the flow of control when an invoked method throws an exception. How to apply this to cases when one is thrown without a try block and from a try block (with appropriate and insufficient exception handlers).

- The difference in calling methods that throw or don't throw exceptions.
- How to recognize the code that can throw these exceptions and handle them appropriately.

Students will be able to:

- [8.3] Describe the advantages of Exception handling.
- [8.1] Differentiate among checked exceptions, unchecked exceptions, and Errors.
- [8.2] Create a try–catch block and determine how exceptions alter normal program flow.
- [8.4] Create and invoke a method that throws an exception.
- [8.5] Recognize common exception classes (such as NullPointerException, Arithmetic– Exception, ArrayIndexOutOfBounds– Exception, ClassCastException)

Resources

Core Text:

OCA Java SE 8 Programmer I Certification Guide (2016). Gupta, Mala (ISBN# 9781617293252)
Oracle Certified Associate Java SE 8 Programmer I (ISBN# ISBN# 9781118957400)

Suggested Resources:

https://education.oracle.com/java–foundations/pexam_1Z0–811
https://education.oracle.com/java–se–8–programmer–i/pexam_1Z0–808