Below is a table of contents for Java, giving definitions, principles, and concepts. Below, **JHT** stands for *JavaHyperText*.

1. Definition of type: **JHT** entry *type*. **JHT** entry primitive *type*.   
   **JHT** entry number *type*. **JHT** entry number *class as a type*.
2. How we draw objects of a class. **JHT** entry *object*.
3. Java wrapper class. **JHT** entry *wrapper*.
4. Overloading method names. **JHT** entry *overload*.
5. Overriding methods, the overriding or bottom-up rule. **JHT** entries *override* and *compile-time reference rule*.
6. Constructor: **JHT** entries *constructor*, *constructor call* and  *constructor call, default*.

* *Main purpose*: Initialize fields of a newly constructed object to make the class invariant true.
* *Principle*: Initialize fields of superclass before fields of subclass
* The first statement of any constructor body you write must be a call on another constructor --**this**(...) or **super**(...). If not, Java inserts **super**().
* If a class C does not have a constructor declared in it, Java declares this one: **public** C(){}

1. What you should know about executing/evaluating:

* The new-expression: What is the syntax of a new-expression? What are the three steps in evaluating a new-expression? Tutorial *Explain constructs/New-expression*.
* In an expression **this**.x or **this**.m(...), what does "**this**" evaluate to? **JHT** entry *this*.
* What are the four steps in executing a method call? **JHT** entry *method call.*
* What’s the syntax of the try-statement? How is a try-statement executed? **JHT** entry *try-statement.*

1. **this**. and **super**. **JHT** entries *this* and *super*.

* In an expression **this**.x or **this**.m(...), what does "**this**" evaluate to? **JHT** entry *this*.
* In an expression **super**.x or **super**.m(...), what does "**super**" evaluate to? **JHT** entry *super*.
* When is this needed "**this**." Only when a variable is shadowed. If that is not the case avoid useless clutter and don’t put in “**this**.”. JavaHyperText entry *shadowing a variable*.

1. Four kinds of variable: field (instance variable), static variable (field), parameter, local variable. **JHT** entry *variable*. Scope of a local variable. **JHT** entry *scope*.
2. Class invariant: **JHT** entry *class invariant* and *Style Guide*, section 3.3.
3. Inside-out rule. **JHT** entry *inside-out rule*.
4. Exception handling. Tutorial *Exceptions*.

* Throwable: the superclass of all throwable objects. **JHT** entry *throwable, class*.
* Subclasses Exception and Error. Tutorial *Exceptions*, section 2, *the throwable object*.
* What a subclass declaration should look like. Tutorial *Exceptions*, section 2, *the throwable object*.
* Execution of a try-statement —you should know the steps. **JHT** entry *try-statement*.
* The throw statement: **JHT** entry *throw-statement*.

1. Abstract classes and methods. Tutorial *Abstract classes and interfaces*.

* Why make a class abstract? Tutorial *Abstract classes and interfaces*, first video.
* Why make a method abstract? Tutorial *Abstract classes and interfaces*, first video

1. Interface. Tutorial *Abstract classes and interfaces*, second video.

* What is the syntax for an interface declaration? Tutorial *Abstract classes and interfaces*, second video.
* How can an interface be used to define a type (at least its syntax, with comments giving meaning). People call these kinds of types ADTs —for Abstract Data Type. **JHT** entry *abstract data type*.

1. Generics. **JHT** entry *generics*.

* Why can’t ArrayList<String> be a subclass (subtype) of ArrayList<Object>? **JHT** entry *abstract data type*.
* Wildcard, bounded wildcard. **JHT** entry *generics*.
* Restrictions on generics. JavaHyperText entry *generics*.