Cracking the New Sun Certified Programmer for Java 2 Platform 1.5 Exam

Khalid Azim Mughal
Department of Informatics
University of Bergen
khalid@ii.uib.no
http://www.ii.uib.no/~khalid

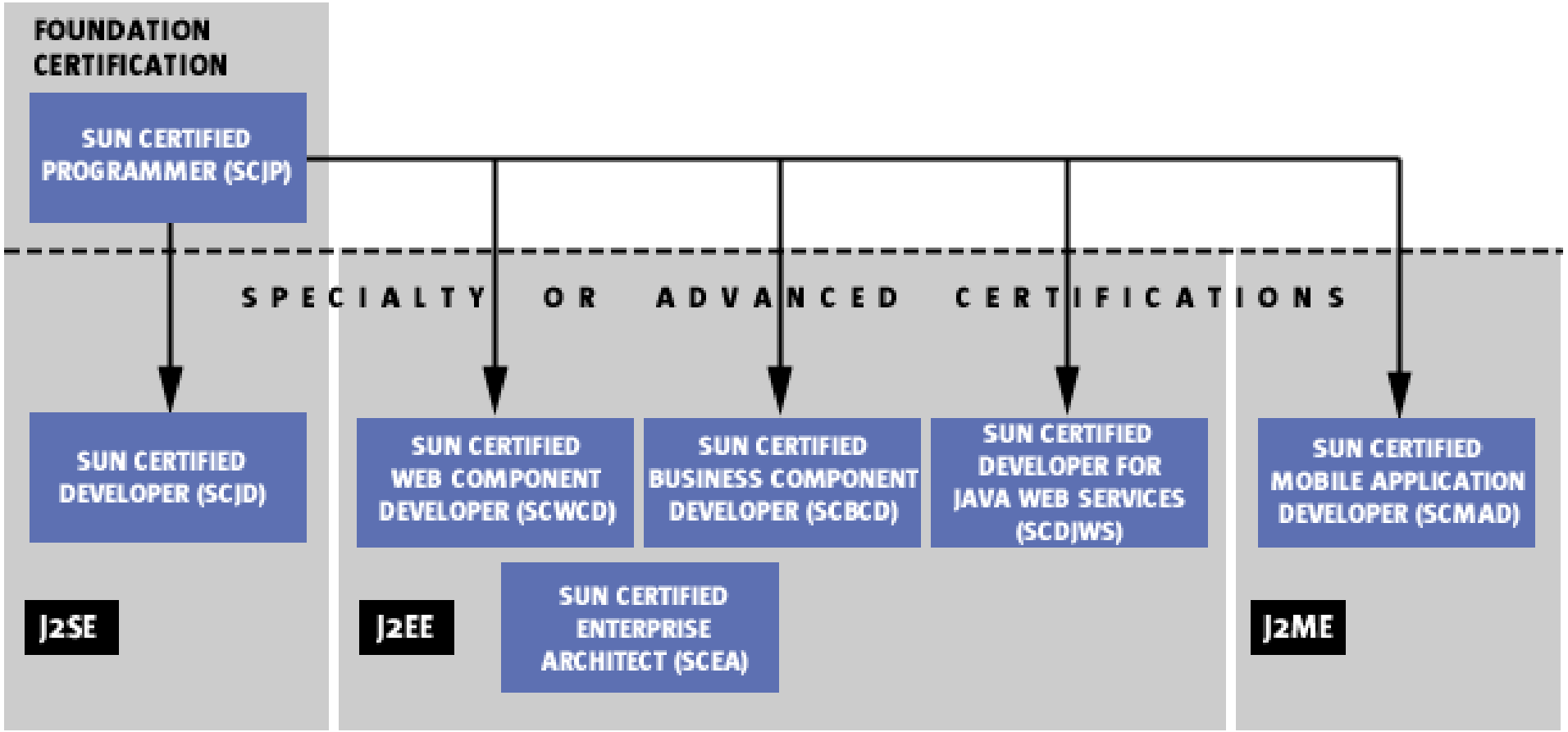
Version date: 2005-09-04

Presentation at JavaZone, Oslo, Norway.
14 September, 2005.

Talk Overview

- Java Technology Certification Overview *(3 min.)*
- Sun Certified Programmer for Java 2 Platform 1.5 (SCJP 1.5) Exam: *(3 min.)*
- Overview of the Objectives of the SCJP 1.5 Exam *(4 min.)*
- Code Scenarios to illustrate the topics on the exam *(30 min.)*
- Preparing for the Exam *(3 min.)*
- Resources for the New Exam *(2 min.)*

Java Technology Certification Overview - 2005



http://www.sun.com/training/certification/java/index.html

Why get certified?

- Making career move from non-technical to application development.
- Criteria for promotion
- Validation of skills
- Improve Java skills
- Certified once, recognized everywhere

SUN CERTIFIED PROGRAMMER FOR JAVA 2 PLATFORM (SCJP)

- **Sun Certified Programmer for the Java 2 Platform, Standard Edition 5.0 (CX-310-055)**
- Sun Certified Programmer for the Java 2 Platform, Standard Edition 5.0 Upgrade Exam (CX-310-056)
- Sun Certified Programmer for the Java 2 Platform, Standard Edition 1.4 (CX-310-035)
- Sun Certified Programmer for the Java 2 Platform, Standard Edition 1.4 Upgrade Exam (CX-310-036)
- Sun Certified Programmer for the Java 2 Platform, Standard Edition 1.2 (CX-310-025)

SCJP 1.5 Exam

Exam No.	**SCJP 1.5 (CX-310-055)**	SCJP 1.4 (CX-310-035)
Prerequisites	**None**	None
Exam Type	**Primarily multiple choice, but also drag and drop**	Multiple choice, short answers
No. of Questions	**72**	61
Passing Score	**59% (43 out of 72)**	52% (32 out of 61)
Time allotted	**175 min.**	120 min.
Total Cost	**USD $150 (or local price)**	USD $150 (or local price)

- Information for taking the exam in Norway:

  https://www.suntrainingcatalogue.com/eduserv/client/welcome.do?l=en_NO
- Contact one of the Thomson Prometric test centers in Norway when you are ready to take the exam.

  http://www.register.prometric.com/Menu.asp
- On passing the exam, Sun will send a certificate by mail.

How the SCJP 1.5 Exam is conducted

- The testing locations: a cubicle with a desktop computer
  - No personal belongings or food allowed.
  - Candidates can make notes, but cannot take the notes with them after the exam.
- Utilizing the allotted time
  - 72 questions within 2 hrs. 55 min., less than 2.5 min. on average to answer each question.
  - If the answer does not become apparent within a reasonable time, move on to the next question.
  - Time permitting, you can return to any questions left unanswered.
  - Review the answers if you have any time remaining.

- The exam program
  - A test run before the actual exam
  - Presents a set of randomly-selected questions one at a time through a graphical user interface.
  - Ability to navigate back and forth through the questions.
  - Use the EXHIBIT-button to open the code scenario in a separate window.
  - After the completion of the exam:
    - An indication of pass or fail
    - The total score
      - Pass score is 59% (43 of 72).
      - Questions weighted equally.
      - No points for partially correct answers.
    - Score on each of the categories of the objectives.
  - The test program will not divulge which questions were answered correctly or the answers to the questions asked.

Types of exam questions: Multiple choice

- Primarily analyzing program code, followed by multiple choices pertaining to the code.
- Multiple true or false statements/code pertaining to a question

Types of answers expected

- Answers to multiple choice questions: select the correct number of choices indicated in the question.
  - All of the appropriate and none of the inappropriate choices must be selected for the question as a whole to be considered correctly answered.
  - Radio button for a single selection, check boxes for multiple selections.
- Another form of questions requires drag-and-drop answers.

SCJP 1.5 Exam Objectives

http://www.sun.com/training/catalog/courses/CX-310-055.xml

- Section 1: Declarations, Initialization and Scoping
- Section 2: Flow Control
- Section 3: API Contents
- Section 4: Concurrency
- Section 5: OO Concepts
- Section 6: Collections / Generics
- Section 7: Fundamentals

Focus on language usage tested through code scenarios,
i.e. be able to read, evaluate, and troubleshoot code.

Objectives in Section 6: Collections / Generics

- Given a design scenario, determine which collection classes and/or interfaces should be used to properly implement that design, including the use of the `Comparable` interface.
- Distinguish between correct and incorrect overrides of corresponding `hashCode` and `equals` methods, and explain the difference between `==` and the `equals` method.
- Write code that uses the generic versions of the Collections API, in particular, the `Set`, `List`, and `Map` interfaces and implementation classes. Recognize the limitations of the non-generic Collections API and how to refactor code to use the generic versions.
- Develop code that makes proper use of type parameters in class/interface declarations, instance variables, method arguments, and return types; and write generic methods or methods that make use of wildcard types and understand the similarities and differences between these two approaches.
- Use capabilities in the `java.util` package to write code to manipulate a list by sorting, performing a binary search, or converting the list to an array. Use capabilities in the `java.util` package to write code to manipulate an array by sorting, performing a binary search, or converting the array to a list. Use the `java.util.Comparator` and `java.lang.Comparable` interfaces to affect the sorting of lists and arrays. Furthermore, recognize the effect of the "natural ordering" of primitive wrapper classes and `java.lang.String` on sorting.

Transition from SCJP 1.4 to SCJP 1.5

Java 1.5 TOPICS	OBJECTIVE
Autoboxing/unboxing	03.01
Covariant `return`	01.05
Enums	01.01, 01.03
`for(:)`-loop	02.02
Generic types, methods, wildcards	06.04
Generic vs. non-generic usage	06.03
`Scanner` class and Regular Expressions (RegEx)	03.05
Static `import`	01.01
`StringBuilder` class	03.01
Var-args	01.04

BEEFED-UP or NEW Java pre-1.5 TOPICS	OBJECTIVE
`Collections` and `Arrays` classes	06.05
`Comparable` and `Comparator` interfaces for wrapper classes	06.05
Constructor and Deserialization	03.03
Exception categorization	02.06
File I/O	03.02, 03.03
Formatting	03.05
Instantiating nested classes	01.06
is-a, has-a relationships	05.05
JAR files, CLASSPATH environment variable	07.05
JavaBeans Standard	01.04
Internationalization: Locales	03.04
Object-oriented (OO)concepts	05.01
Programmatic Garbage Collection (GC)	07.04

TOPICS OMITTED
Low-level (lexical) and esoteric language details
`Bitwise operators`
`Shift operators`
`Math` class

SCJP 1.5 Exam Topics by Code Scenarios

Using Generics, etc.

Q1 Given:

```
import java.util.*;
class Belt {
  int size;
  Belt(int s) { size = s; }
  static Comparator________ makeComparator() {
    return new ________ ________() {
        public int _________ (Belt b1, Belt b2) { return __________ - _________; }
    };
  }
}

public class CraftingBelts {
  public static void main(String[] args) {
    List________ bList = new ArrayList ________ ();
    bList.add(new Belt(38)); bList.add(new Belt(30));
    bList.add(new Belt(25)); bList.add(new Belt(42));

    Comparator ________ comp = ________.makeComparator();
    Collections.________( ________, ________ );
    for ( ________ b : ________ )
      System.out.print( ________ + " ");
  }
}
```

Fill in the blanks in the above code so that it compiles without errors and warnings, and prints "42 38 30 25 ". Only the following code fragments can be used, multiple times if necessary, to fill in the blanks.

```
<Belt>     <CraftingBelts>   <Comparator>
Belt       CraftingBelts     Comparator
compare    compareTo         sort          sortList
b1         b2                b             bList
b1.size    b2.size           b.size        bList.size()  bList.values()
```

Subtyping and Generics

Q2 Which of the following declarations will compile without errors or warnings, assuming all necessary types have been imported? Select the 2 correct answers.

```
(a)  ArrayList<Object> aList = new ArrayList<String>();
(b)  List<String> bList = new ArrayList<String>();
(c)  List<Object> cList = new ArrayList<String>();
(d)  List<?> dList = new ArrayList<String>();
(e)  List<? extends String> eList = new ArrayList<String>();
(f)  List<? extends Comparable> fList = new ArrayList<String>();
(g)  List<? extends Object> gList = new ArrayList<String>();
(h)  List<? super String> hList = new ArrayList<String>();
(i)  List<? super Comparable> iList = new ArrayList<String>();
(j)  List<? super Object> jList = new ArrayList<String>();
```

Mixed Bag of Generics

Q3 Given:

```
class MixBag<T extends Comparable<Integer> & Comparable<String>> {
  public static     void do1() {T t; }
  public static <T> void do2() {T t; }
  public int compareTo(Integer f) { return 0; }
  public int compareTo(String f) { return 0; }
  public <T extends Exception>
    void handleException(T t)
      throws T {
    try { throw t; }
    catch (T e) { }
  }
}
```

The compile flags errors at which lines in the above code? Select the 3 correct answer.

(a) At (1)
(b) At (2)
(c) At (3)
(d) At (4)
(e) At (5)
(f) At (6)
(g) At (7)
(h) At (8)

Arrays and Generics

Q4 Given:

```
class Box<T> {
  private T data;
  Box(T d) { data = d; }
  public T getData() { return data; }
 }
public class ArraysAndGenerics {
  public static void main(String[] args) {
    // (1) Insert code line here
    addElements(boxes);
    Box<Integer> b = boxes[1];       // (2)
    Integer i = b.getData();         // (3)
  }
  public static void addElements(Object[] objArr) {
    objArr[0] = new Box<Integer>(100);
    objArr[1] = new Box<Integer>(200);
  }
}
```

Which declaration when inserted independently at (1) will result in the above code to compile and execute normally? Select the one correct answer.

(a) `Box[] boxes = new Box[2];`

(b) `Box<?>[] boxes = new Box<?>[2];`

(c) `Box<Integer>[] boxes = new Box<Integer>[2];`

Casting and Generics

Q5 Given:

```
class Item<T> {
  private T data;
  Item(T d) { data = d; }

  public boolean equals(Object other) {
    if (this == other) return true;
    if (other == null) return false;
    if (this.getClass() != other.getClass()) return false;
    // (1) Insert code here.
    return (this.data.equals(otherItem.data));
  }
}
```

Which method declaration when inserted at (1) will result in the above code to compile without errors and warnings? Select the one correct answer.

(a) `Item<?> otherItem = (Item<?>) other;`

(b) `Item<T> otherItem = (Item<T>) other;`

(c) `Item<T> otherItem = (Item<?>) other;`

(d) `Item<?> otherItem = (Item<T>) other;`

instanceof and Generics

Q6 Given:

```
import java.util.*;

class ForInstance {
  public static void main(String[] args) {
       Object o = new HashSet<Double>();
       // (1) Insert code here.
  }
}
```

Which code when inserted at (1) will result in the program to compile and execute normally? Select the one answer.

(a) `System.out.println (o instanceof Set);`

(b) `System.out.println (o instanceof Set<?>);`

(c) `System.out.println (o instanceof Set<Double>);`

(d) `System.out.println (o instanceof Set<? extends Number>);`

(e) `System.out.println (o instanceof Set<? super Number>);`

Generic Method Invocation

Q7 Given:

```
class GMI {
  private static <T> void genericStaticMethod()    {  }
  private        <T> void genericInstanceMethod()  {  }

  public static void main(String[] args) { new GMI().doCalling(); }

  public void doCalling() {
    // (1) Insert code here.
  }
}
```

Which method declaration when inserted at (1) will result in the above code to compile without errors and warnings? Select the one answer.

(a) `GMI.genericStaticMethod();`
(b) `<String>genericStaticMethod();`
(c) `GMI.<String>genericStaticMethod ();`
(d) `genericInstanceMethod();`
(e) `this.genericInstanceMethod();`
(f) `<String>genericInstanceMethod();`
(g) `this.<String>genericInstanceMethod ();`

Some Facts about Generics

Q8 Which statements are true about generics in Java? Select the 2 correct answers.

(a) Static member classes and interfaces can have type parameters.
(b) Non-static member classes can have type parameters.
(c) Anonymous classes cannot have type parameters.
(d) Exception types can have type parameters.
(e) Exception types can be used as type parameter bounds.
(f) Enum type can have type parameters.
(g) Parameterized types `ArrayList<String>` and `ArrayList<Long>` have different types at compile time, but share the same runtime type, i.e. `ArrayList`.
(h) Arrays of concrete parameterized types cannot be created, i.e. `new ArrayList<String>[2]`.
(i) Arrays of unbounded wildcard parameterized types can be created, i.e. `new ArrayList<?>[2]`.
(j) Arrays of raw types can be created, i.e. `new ArrayList[2]`.

Enums

Q9 Given:

```
enum Meal {
    BREAKFAST(8),
    // (1) Insert code here.
    DINNER(18);
    Meal(int hh) { this.hh = hh; }
    private int hh;
    public int getHour() { return this.hh; }
    public void setHour(int hh) { this.hh = hh; }
}
public class MealClient {
  public static void main(String[] args) {
      System.out.println(Meal.LUNCH + " served at " + Meal.LUNCH.getHour());
  }
}
```

Which code when inserted at (1) will result in the program to compile and print `"LUNCH served at 13"`? Select the one correct answer.

(a) `LUNCH(0) { hh = 13; },`

(b) `LUNCH(0) { this.hh = 13; },`

(c) `LUNCH(0) { this.setHour(13); },`

(d) `LUNCH(0) { public int getHour() { return (BREAKFAST.hh + DINNER.hh)/2; } },`

(e) None of the above code will produce the specified output.

Covariant return

Q10 Given:

```
class A {}
class B extends A {}
class C extends B {}

class X           { public B getIt() {return new B();} }
class Y extends X {
  // (1) Insert code here.
}
```

Which methods when inserted independently at (1) will cause the above code to compile without errors or warnings? Select the 2 correct answers.

(a) `public C getIt() { return new C(); }`
(b) `public B getIt() { return new C(); }`
(c) `public A getIt() { return new C(); }`
(d) `public A getIt() { return new B(); }`
(e) `public C getIt() { return new B(); }`

Auto Boxing and Unboxing

Q11 Given:

```
class Wrapping {
    public static void main(String[] args) {
        Character[] version = {'1', '.', '5'};                    // (1)
        for (Integer iRef = 0;                                    // (2)
             iRef < version.length;                               // (3)
             ++iRef)                                              // (4)
            switch(iRef) {                                        // (5)
              case 0: case 2:
                System.out.println(iRef + ": " + version[iRef]); // (6)
            }
    }
}
```

Which statement is true about the program?

(a) The values in the array initialization block at (1) are not assignable to elements of a `Character` array.
(b) The operands in the expression `iRef < version.length` at (3) have incompatible types.
(c) The operand of the ++ operator at (4) cannot be applied to a value of `Integer` type.
(d) The type of the `switch` expression at (5) cannot be `Integer`.
(e) The type of the index in the expression `version[iRef]` at (6) cannot be `Integer`.
(f) The above code will compile and print:

```
0: 1
2: 5
```

(g) None of the above.

for(:) loop

Q12 Given:

```
import java.util.*;
import static java.lang.System.out;
public class Looping {
  public static void main(String[] args) {
    List<Double> versions = new ArrayList<Double>();
    versions.add(1.0); versions.add(1.5); versions.add(2.0);
    for (Double v : versions.values()) {
      v = v + 0.5;
    }
    out.println(versions);
  }
}
```

Which statement is true about the program?

(a) The code will compile and print: [1.5, 2.0, 2.5]

(b) The code will compile and print: [1.0, 1.5, 2.0]

(c) The code will not compile.

Var-args

Q13 Given:

```
class Arrrrg {
    public static void main(String... args) {
        doIt(1);
        doIt(1,2);
    }
    // (1) Insert code here.
}
```

Which method declaration when inserted independently at (1) will result in the above code to compile and execute normally? Select the two correct answer.

(a) `static void doIt(int... params) { }`
(b) `static void doIt(int[] params) { }`
(c) `static void doIt(int params...) { }`
(d) `static void doIt(int... params, int i) { }`
(e) `static void doIt(int j, int... params) { }`
(f) `static void doIt(int j, int... params[]) { }`
(g) The code will not compile regardless of which method declaration is inserted.

SCJP 1.5 Exam Preparation

- Real-world experience alone is not enough!
- Just theory is also not enough!
- Emphasis on language *usage* through code scenarios: *read, evaluate and troubleshoot*.
- Preparation comprises:
  - Study relevant material on the exam objectives
  - Take notes
  - Experiment with code
  - Join an online discussion group (*see below*)
  - Practice taking mock exams

Must-have list for SCJP 1.5 Exam Preparation

- One stop for exam resources: *Javaranch.com*
  http://saloon.javaranch.com/cgi-bin/ubb/ultimatebb.cgi?ubb=forum&f=24
- Few books covering the exam topics at the moment.
- JDK 1.5 with API documentation
- Download mock exams

RESOURCES

1. James Gosling, Bill Joy, Guy Steele, Gilad Bracha: *The Java Language Specification*, 3/e, Addison-Wesley Professional, 2005, ISBN: 0321246780

   See also JLS Maintenance Page at *http://java.sun.com/docs/books/jls/jls-maintenance.html*.
2. Gilad Bracha. Generics in the Java Programming Language, July 2004. (*http://java.sun.com/j2se/1.5/pdf/generics-tutorial.pdf*)
3. Ken Arnold, James Gosling, David Holmes: *Java™ Programming Language*, 4/e, Addison Wesley Professional, 2006, ISBN: 0-321-34980-6
4. Katherine Sierra, Bert Bates: *SCJP Sun Certified Programmer for Java 2 Platform 5 Study Guide* (Exam 310-055), McGraw-Hill Osborne , 2005, ISBN: 0072253606

Good luck!

You will need it!

The exam is not a joke!

Comments? Questions? Suggestions?

:-)