



AAS BRIEF TECHNICAL REPORT (10 – 008):

Comparison of Object-Oriented Languages in the Implementation of Smart Test Technology ®: Part III

Damon Bryant, PhD, Adaptive Assessment Services

This research is an extension of studies that document the transition of the Smart Test Technology ® platform from its current implementation in Python to a potentially faster implementation in Java. The previous studies are titled Comparison of Object-Oriented Languages in the Implementation of Smart Test Technology ®: Parts I and II. Part I investigated different object-oriented implementations of time-sensitive components related to the creation and scoring of test questions. Results of Part I indicated that Java is approximately 4 times faster than Python in creating and scoring test questions. Part II investigated the estimation of examinee ability using dimensionally complex test questions. Results of Part II showed that Java is approximately 200 times faster than Python in estimating an examinee’s ability.

The purpose of this study is to compare the implementation of another time-sensitive component within the Smart Test Technology® platform: computer adaptive item administration with content constraints. This time-sensitive component involves source code examined in Parts I and II. Specifically, several objects related to aspects of item creation, item scoring, and examinee ability estimation are needed to evaluate the computer adaptive algorithm. The comparison is made between Python 2.6 and Java 6.0 in terms of operational runtimes. The source code is proprietary and further details about the code are not provided.

This research compares language execution times of approximately 20,000 calculations to estimate an examinee’s ability to a 7-item computer adaptive test. The test items are selected and administered from an available pool of 16-items. The process flow of execution is as follows: (1) an examinee is simulated with a randomly generated ability level θ , (2) the item pool of 16 questions is created, (3) the simulated examinee is administered a test question



based upon the initial estimate of ability θ , (4) the question is answered by the simulated examinee using a short algorithm, (5) the simulated examinee's ability is estimated along with the error in the estimate based on the response to the question, (6) the next item is adaptively selected and administered based upon the new ability estimate and content constraints of the overall test, (7) steps 3 through 6 are repeated until seven items have been administered, and (8) a final ability estimate and the error in the ability estimate are computed.

There are several independent variables of interest in this research: (1) programming language, (2) item administration method, and (3) content constraints algorithm. The administration methods include Smart Test Technology[®] proprietary algorithm (STT), random item selection (RAN), and maximum information (MAX). MAX is the traditional criterion used to select questions in operational computer adaptive testing programs such as the SAT administered by the Educational Testing Service and the ASVAB administered by the US Armed Forces. The efficiency of STT over MAX has been documented elsewhere; See Smart Test Technology[®] as a Computer Adaptive Item Selection Algorithm (Bryant, 2010). A content constraints algorithm used to restrict item administration based upon pre-defined criteria is evaluated with two levels: content constraints imposed versus no content constraints imposed. The impact of content constraints on test scores is documented in Validation of Content Constraints for the Computer Adaptive Abstract Reasoning Test (Bryant, 2010).

After implementing several classes in both languages, the execution of the code was timed. The methods involved over 20,000 calculations using a variety of classes. The language variable was crossed with each of the two variables to create two ANOVA models: Language X Method (2 x 3) and Language X Content Constraints (2 x 2) between-subjects designs. Each condition was replicated 20 times for a total of 240 trials (2x3x2x20). Each runtime was recorded for analysis. As in Part II, it was verified that the two language implementations produced the exact same results in Java and Python. With the equality of output established, the independent variables were the programming language used, item administration method,



and content constraints imposed. The dependent variable was the runtime of the executed code recorded in milliseconds.

Across 240 trials, the average runtime for completing a seven-item computer adaptive test in Python 2.6 was 1459 milliseconds or 1.5 seconds. Across 240 trials, the average runtime in Java 6.0 was 74 milliseconds or 0.07 seconds. See Figure 1. Results of an independent t-test (unequal variance assumed) indicate that the difference in mean runtimes is statistically significant, $t(120) = 116.7, p < .0001$. The runtime in Java was approximately 19.7 times faster than the runtime in Python.

With respect to item administration method within Python, the average runtime of STT adaptive item administration was 1414 milliseconds or 1.4 seconds. The average runtime of RAN item selection was also 1414 milliseconds (1.4 seconds), and the runtime of MAX item selection was 1550 milliseconds (1.6 seconds). With respect to item administration method within Java, the average runtime of STT adaptive item administration was 79 milliseconds or .08 seconds. The average runtime of RAN item selection was 73 milliseconds (.07 seconds), and the runtime of MAX item selection was 70 milliseconds (.07 seconds). The overall model for the Language (2 levels) X Administration Method (3 levels) ANOVA was statistically significant, $F(5,234) = 3556.2, p < .0001$, accounting for 99% of the variance in runtime. See Figure 2.

In regard to the average runtime of content constraints in Python, the no constraints condition was 1404 milliseconds (1.4 seconds), and the runtime of the content constraints imposed condition was 1515 milliseconds (1.5 seconds). In reference to the mean runtime of content constraints in Java, the no content constraints condition had an average runtime of 75 milliseconds and the runtime of the content constraints imposed condition was 73 milliseconds. This 2x3 ANOVA model was statistically significant $F(3,236) = 5530, p < .0001$, accounting for 99% of the variance in runtime.

In summary, the most significant variable to have an impact on runtime performance was language implementation. Under the conditions in this study, Java is recommended in



terms of runtime performance as compared to Python. In Part I, the results indicated that Java was almost 4 times faster than Python in creating and scoring a test question. In Part II, the results showed that Java was approximately 200 times faster than Python in estimating an examinee's ability. In this final investigation of time-sensitive components within Smart Test Technology[®], the results of the statistical analysis indicates that Java is approximately 20 times faster in operational performance as compared to Python in administering a short computer adaptive test. This difference is statistically and practically significant. Although Python was very useful in the implementation of the Smart Test Technology[®] prototype, the evidence in this study along with the results of previous comparisons of Java indicates that Java will provide a much faster user experience in the administration of an artificially intelligent platform for the assessment of human capabilities.



Figure 1. Graph of Average Runtime of Java and Python in Milliseconds across 240 Trials

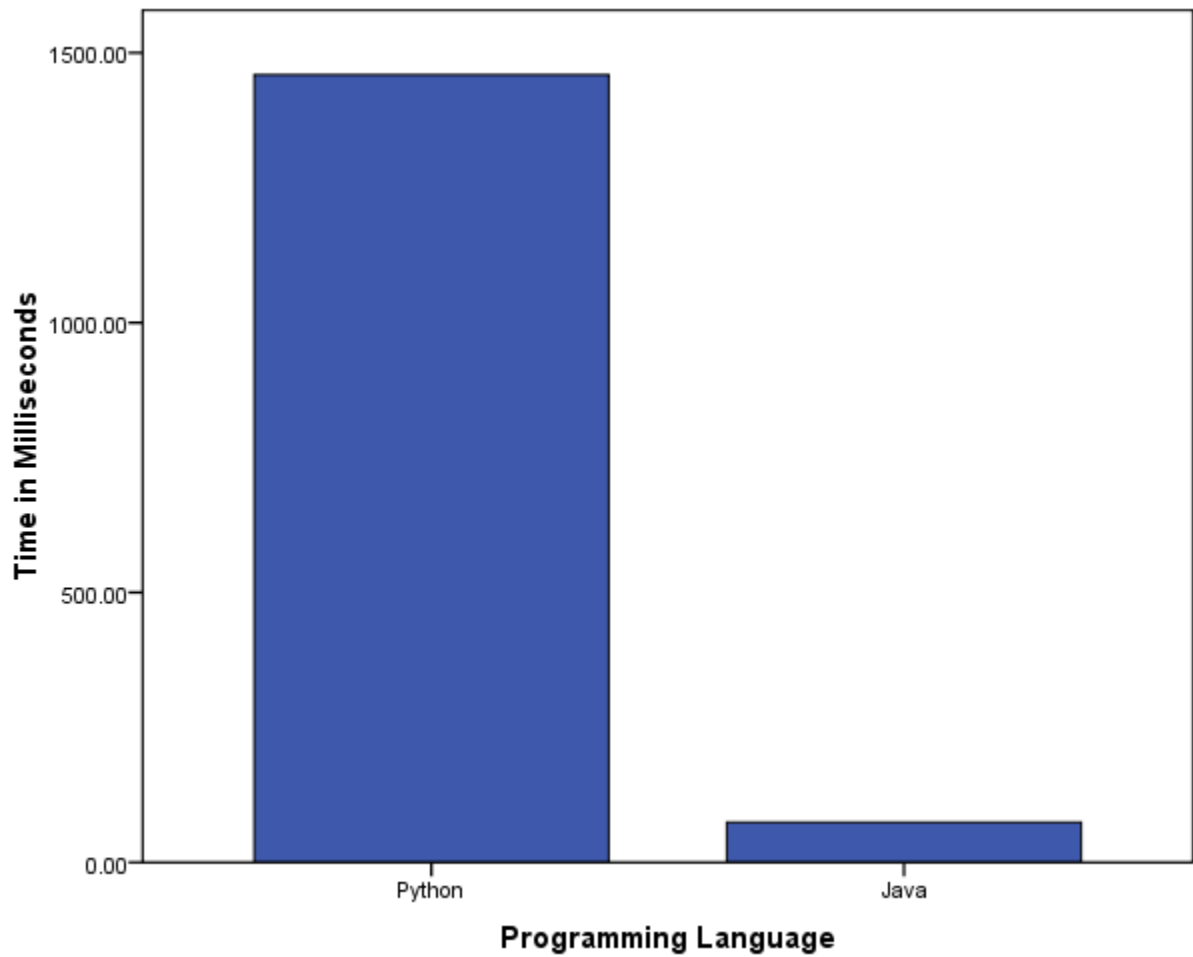




Figure 2. Graph of Average Runtime of Item Selection Methods in Milliseconds across 240 Trials

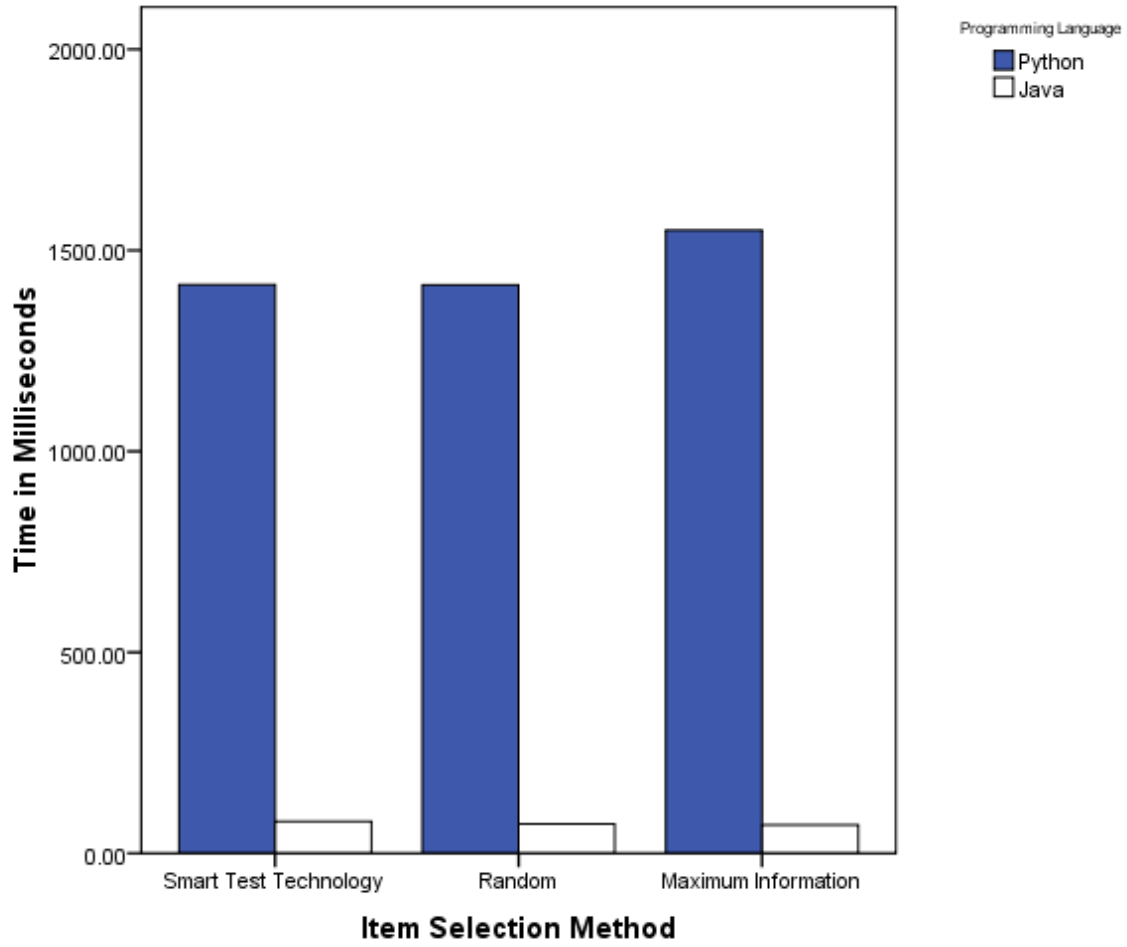




Figure 3. Graph of Average Runtime of Content Constraints in Milliseconds across 240 Trials

