# COBOL and JCL to Java and Python Deutsche Bank — KreditManager, OpenShift Platform Applications

| CLIENT | SOFTWARE | LANGUAGE PAIRING | COMPLETION TIME |
|---|---|---|---|
| Deutsche Bank | KreditManager, OpenShift Platform Applications | COBOL and JCL to Java and Python | 12 months |

## EXECUTIVE SUMMARY

With advances in digital banking over the past two decades, Deutsche Bank, a global banking company with a presence in 58 countries, saw it lacked the agility to compete with the speed and easy access of the many mobile finance apps on the market. Deutsche Bank's leaders knew they would need to be cloud-enabled with a modern architecture to stay relevant for its customers and ongoing market needs.

## HIGHLIGHTS

- **Near-100% Automation**
- **Low Technical Risk**
- **Step-Wise Methodology**

## PROCESS

Deutsche Bank selected TSRI to run a pilot project that would transform two banking applications: EZB DepotStatistik and DBSAE. Both transformations modernized existing COBOL applications to Java, which would allow them to run on Deutsche Bank's private OpenShift cloud. TSRI then transformed the rest of Deutsche Bank's legacy applications and migrated two databases. Using a step-wise methodology, which brought smaller applications online first and successively modernized more over time to avoid long disruptions, TSRI migrated 383,358 lines of COBOL code and 13,864 lines of JCL code to a multi-tier environment at 99.7% automation. Oracle served as the new database environment.

Following the modernization, TSRI's iterative refactoring phase utilized results from the SonarQube code-quality tool to develop refactoring rules to improve and remediate issues. In one instance, TSRI's automated code saved the client approximately 12,500 hours in software development time.

## WHY MODERNIZE WITH TSRI?

- **Preserve business logic.** A model-based approach preserves the source business logic.
- **Reduce code freeze to as little as a few days.** Automation enables continued development and baselines can automatically be taken at any time.
- **Improve code quality.** Pattern-based refactoring applied to the entire code base improves code maintainability, reduces technical debt, and remediates security flaws inherent in legacy code.
- **Save money.** Near-100-percent automation permits economies of scale when modernizing larger applications while supporting fast and cost-effective customization of a single project.