Dev Challenge (Short)

The Asset Management Digital Challenge

We are providing you with a simple REST service with some very basic functionality - to add and read an account.

It is a standard gradle project, running on Spring Boot. It uses Lombok and if you've not come across it before you'll need to configure your IDE to use it (otherwise code will not compile).

Your task is to **add functionality for a transfer of money between accounts**. Transfers should be specified by providing:

- accountFrom id
- accountTo id
- amount to transfer between accounts

The amount to transfer should always be a positive number.
It should not be possible for an account to end up with negative balance (we do not support overdrafts!)

Whenever a transfer is made, a notification should be sent to both account holders, with a message containing id of the other account and amount transferred.

- For this purpose please use the NotificationService interface
- **Do NOT** provide implementation for this service - it is assumed another colleague would implement it.
- Do not use the provided (simple) implementation in your tests - it is provided for the main application to run. In your tests you should mock this service.

This feature should be implemented in a thread-safe manner. Your solution should never deadlock, should never result in corrupted account state, and should work efficiently for multiple transfers happening at the same time.

Additional guidance

Please provide us with the code, preferably via github.
We think this task should take you about an hour - take the **simplest possible approach that works**. It's not meant to be a trick question, and our expectations are set accordingly to the time we expect you to spend on this.

Please treat this as an opportunity to showcase how you work, quality of what you provide will matter much more than pure quantity of code, or adding features we do not ask for.
Clean, elegant and simple code wins over feature rich every time.

Due to limited time we expect you to spend on this, we are happy for you to focus only on the code and tests to deliver the feature.
However, please provide a **short** document (a few bullet points will suffice) describing any extra work you would consider important to do before this project was turned into a production application - i.e. what would you improve/add, given more time.

Please focus on the application being "production-and-support-ready" for requirements already provided - not on extra features/functionality that could be added

Gradle Runner

Unfortunately due to restrictions on what can be sent from DB, we had to remove the gradlew runner from this project. Therefore, you will need to run it via Gradle daemon (use latest version). Alternatively, you can re-add gradlew.bat and gradle wrapper jar from another project.

gradlew

```
#!/usr/bin/env sh
##############################################################################
##
##  Gradle start up script for UN*X
##
##############################################################################
# Attempt to set APP_HOME
# Resolve links: $0 may be a link
```

```
PRG="$0"
# Need this for relative symlinks.
while [ -h "$PRG" ] ; do
    ls=`ls -ld "$PRG"`
    link=`expr "$ls" : '.*-> \(.*\)$'`
    if expr "$link" : '/.*' > /dev/null; then
        PRG="$link"
    else
        PRG=`dirname "$PRG"`"/$link"
    fi
done
SAVED="`pwd`"
cd "`dirname \"$PRG\"`/" >/dev/null
APP_HOME="`pwd -P`"
cd "$SAVED" >/dev/null
APP_NAME="Gradle"
APP_BASE_NAME=`basename "$0"`
# Add default JVM options here. You can also use JAVA_OPTS and GRADLE_OPTS to pass
JVM options to this script.
DEFAULT_JVM_OPTS=""
# Use the maximum available, or set MAX_FD != -1 to use that value.
MAX_FD="maximum"
warn ( ) {
    echo "$*"
}
die ( ) {
    echo
    echo "$*"
    echo
    exit 1
}
# OS specific support (must be 'true' or 'false').
cygwin=false
msys=false
darwin=false
nonstop=false
case "`uname`" in
  CYGWIN* )
    cygwin=true
    ;;
  Darwin* )
    darwin=true
    ;;
  MINGW* )
    msys=true
    ;;
  NONSTOP* )
    nonstop=true
    ;;
esac
CLASSPATH=$APP_HOME/gradle/wrapper/gradle-wrapper.jar
# Determine the Java command to use to start the JVM.
if [ -n "$JAVA_HOME" ] ; then
    if [ -x "$JAVA_HOME/jre/sh/java" ] ; then
        # IBM's JDK on AIX uses strange locations for the executables
        JAVACMD="$JAVA_HOME/jre/sh/java"
    else
        JAVACMD="$JAVA_HOME/bin/java"
    fi
    if [ ! -x "$JAVACMD" ] ; then
        die "ERROR: JAVA_HOME is set to an invalid directory: $JAVA_HOME
Please set the JAVA_HOME variable in your environment to match the
location of your Java installation."
```

```
    fi
else
    JAVACMD="java"
    which java >/dev/null 2>&1 || die "ERROR: JAVA_HOME is not set and no 'java'
command could be found in your PATH.
Please set the JAVA_HOME variable in your environment to match the
location of your Java installation."
fi
# Increase the maximum file descriptors if we can.
if [ "$cygwin" = "false" -a "$darwin" = "false" -a "$nonstop" = "false" ] ; then
    MAX_FD_LIMIT=`ulimit -H -n`
    if [ $? -eq 0 ] ; then
        if [ "$MAX_FD" = "maximum" -o "$MAX_FD" = "max" ] ; then
            MAX_FD="$MAX_FD_LIMIT"
        fi
        ulimit -n $MAX_FD
        if [ $? -ne 0 ] ; then
            warn "Could not set maximum file descriptor limit: $MAX_FD"
        fi
    else
        warn "Could not query maximum file descriptor limit: $MAX_FD_LIMIT"
    fi
fi
# For Darwin, add options to specify how the application appears in the dock
if $darwin; then
    GRADLE_OPTS="$GRADLE_OPTS \"-Xdock:name=$APP_NAME\"
\"-Xdock:icon=$APP_HOME/media/gradle.icns\""
fi
# For Cygwin, switch paths to Windows format before running java
if $cygwin ; then
    APP_HOME=`cygpath --path --mixed "$APP_HOME"`
    CLASSPATH=`cygpath --path --mixed "$CLASSPATH"`
    JAVACMD=`cygpath --unix "$JAVACMD"`
    # We build the pattern for arguments to be converted via cygpath
    ROOTDIRSRAW=`find -L / -maxdepth 1 -mindepth 1 -type d 2>/dev/null`
    SEP=""
    for dir in $ROOTDIRSRAW ; do
        ROOTDIRS="$ROOTDIRS$SEP$dir"
        SEP="|"
    done
    OURCYGPATTERN="(^($ROOTDIRS))"
    # Add a user-defined pattern to the cygpath arguments
    if [ "$GRADLE_CYGPATTERN" != "" ] ; then
        OURCYGPATTERN="$OURCYGPATTERN|($GRADLE_CYGPATTERN)"
    fi
    # Now convert the arguments - kludge to limit ourselves to /bin/sh
    i=0
    for arg in "$@" ; do
        CHECK=`echo "$arg"|egrep -c "$OURCYGPATTERN" -`
        CHECK2=`echo "$arg"|egrep -c "^-"`                                 ###
Determine if an option
        if [ $CHECK -ne 0 ] && [ $CHECK2 -eq 0 ] ; then                    ###
Added a condition
            eval `echo args$i`=`cygpath --path --ignore --mixed "$arg"`
        else
            eval `echo args$i`="\"$arg\""
        fi
        i=$((i+1))
    done
    case $i in
        (0) set -- ;;
        (1) set -- "$args0" ;;
        (2) set -- "$args0" "$args1" ;;
```

```
        (3) set -- "$args0" "$args1" "$args2" ;;
        (4) set -- "$args0" "$args1" "$args2" "$args3" ;;
        (5) set -- "$args0" "$args1" "$args2" "$args3" "$args4" ;;
        (6) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" ;;
        (7) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" "$args6"
;;
        (8) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" "$args6"
"$args7" ;;
        (9) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" "$args6"
"$args7" "$args8" ;;
    esac
fi
# Escape application args
save ( ) {
    for i do printf %s\\n "$i" | sed "s/'/'\\\\''/g;1s/^/'/;\$s/\$/' \\\\/" ; done
    echo " "
}
APP_ARGS=$(save "$@")
# Collect all arguments for the java command, following the shell quoting and
substitution rules
eval set -- $DEFAULT_JVM_OPTS $JAVA_OPTS $GRADLE_OPTS
"\"-Dorg.gradle.appname=$APP_BASE_NAME\"" -classpath "\"$CLASSPATH\""
org.gradle.wrapper.GradleWrapperMain "$APP_ARGS"
# by default we should be in the correct project dir, but when run from Finder on
Mac, the cwd is wrong
if [ "$(uname)" = "Darwin" ] && [ "$HOME" = "$PWD" ]; then
```

```
  cd "$(dirname "$0")"
fi
exec "$JAVACMD" "$@"
```

gradle.bat

```
#!/usr/bin/env sh
##############################################################################
##
##  Gradle start up script for UN*X
##
##############################################################################
# Attempt to set APP_HOME
# Resolve links: $0 may be a link
PRG="$0"
# Need this for relative symlinks.
while [ -h "$PRG" ] ; do
    ls=`ls -ld "$PRG"`
    link=`expr "$ls" : '.*-> \(.*\)$'`
    if expr "$link" : '/.*' > /dev/null; then
        PRG="$link"
    else
        PRG=`dirname "$PRG"`"/$link"
    fi
done
SAVED="`pwd`"
cd "`dirname \"$PRG\"`/" >/dev/null
APP_HOME="`pwd -P`"
cd "$SAVED" >/dev/null
APP_NAME="Gradle"
APP_BASE_NAME=`basename "$0"`
# Add default JVM options here. You can also use JAVA_OPTS and GRADLE_OPTS to pass
JVM options to this script.
DEFAULT_JVM_OPTS=""
# Use the maximum available, or set MAX_FD != -1 to use that value.
MAX_FD="maximum"
warn ( ) {
    echo "$*"
}
die ( ) {
    echo
    echo "$*"
    echo
    exit 1
}
# OS specific support (must be 'true' or 'false').
cygwin=false
msys=false
darwin=false
nonstop=false
case "`uname`" in
  CYGWIN* )
    cygwin=true
    ;;
  Darwin* )
    darwin=true
    ;;
```

```
  MINGW* )
    msys=true
    ;;
  NONSTOP* )
    nonstop=true
    ;;
esac
CLASSPATH=$APP_HOME/gradle/wrapper/gradle-wrapper.jar
# Determine the Java command to use to start the JVM.
if [ -n "$JAVA_HOME" ] ; then
    if [ -x "$JAVA_HOME/jre/sh/java" ] ; then
        # IBM's JDK on AIX uses strange locations for the executables
        JAVACMD="$JAVA_HOME/jre/sh/java"
    else
        JAVACMD="$JAVA_HOME/bin/java"
    fi
    if [ ! -x "$JAVACMD" ] ; then
        die "ERROR: JAVA_HOME is set to an invalid directory: $JAVA_HOME
Please set the JAVA_HOME variable in your environment to match the
location of your Java installation."
    fi
else
    JAVACMD="java"
    which java >/dev/null 2>&1 || die "ERROR: JAVA_HOME is not set and no 'java'
command could be found in your PATH.
Please set the JAVA_HOME variable in your environment to match the
location of your Java installation."
fi
# Increase the maximum file descriptors if we can.
if [ "$cygwin" = "false" -a "$darwin" = "false" -a "$nonstop" = "false" ] ; then
    MAX_FD_LIMIT=`ulimit -H -n`
    if [ $? -eq 0 ] ; then
        if [ "$MAX_FD" = "maximum" -o "$MAX_FD" = "max" ] ; then
            MAX_FD="$MAX_FD_LIMIT"
        fi
        ulimit -n $MAX_FD
        if [ $? -ne 0 ] ; then
            warn "Could not set maximum file descriptor limit: $MAX_FD"
        fi
    else
        warn "Could not query maximum file descriptor limit: $MAX_FD_LIMIT"
    fi
fi
# For Darwin, add options to specify how the application appears in the dock
if $darwin; then
    GRADLE_OPTS="$GRADLE_OPTS \"-Xdock:name=$APP_NAME\"
\"-Xdock:icon=$APP_HOME/media/gradle.icns\""
fi
# For Cygwin, switch paths to Windows format before running java
if $cygwin ; then
    APP_HOME=`cygpath --path --mixed "$APP_HOME"`
    CLASSPATH=`cygpath --path --mixed "$CLASSPATH"`
    JAVACMD=`cygpath --unix "$JAVACMD"`
    # We build the pattern for arguments to be converted via cygpath
    ROOTDIRSRAW=`find -L / -maxdepth 1 -mindepth 1 -type d 2>/dev/null`
    SEP=""
    for dir in $ROOTDIRSRAW ; do
        ROOTDIRS="$ROOTDIRS$SEP$dir"
        SEP="|"
    done
    OURCYGPATTERN="(^($ROOTDIRS))"
    # Add a user-defined pattern to the cygpath arguments
    if [ "$GRADLE_CYGPATTERN" != "" ] ; then
```

```
        OURCYGPATTERN="$OURCYGPATTERN|($GRADLE_CYGPATTERN)"
    fi
    # Now convert the arguments - kludge to limit ourselves to /bin/sh
    i=0
    for arg in "$@" ; do
        CHECK=`echo "$arg"|egrep -c "$OURCYGPATTERN" -`
        CHECK2=`echo "$arg"|egrep -c "^-"`                                  ###
Determine if an option
        if [ $CHECK -ne 0 ] && [ $CHECK2 -eq 0 ] ; then                     ###
Added a condition
            eval `echo args$i`=`cygpath --path --ignore --mixed "$arg"`
        else
            eval `echo args$i`="\"$arg\""
        fi
        i=$((i+1))
    done
    case $i in
        (0) set -- ;;
        (1) set -- "$args0" ;;
        (2) set -- "$args0" "$args1" ;;
        (3) set -- "$args0" "$args1" "$args2" ;;
        (4) set -- "$args0" "$args1" "$args2" "$args3" ;;
        (5) set -- "$args0" "$args1" "$args2" "$args3" "$args4" ;;
        (6) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" ;;
        (7) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" "$args6"
;;
        (8) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" "$args6"
"$args7" ;;
        (9) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" "$args6"
"$args7" "$args8" ;;
    esac
fi
# Escape application args
save ( ) {
    for i do printf %s\\n "$i" | sed "s/'/'\\\\''/g;1s/^/'/;\$s/\$/' \\\\/" ; done
    echo " "
}
APP_ARGS=$(save "$@")
# Collect all arguments for the java command, following the shell quoting and
substitution rules
eval set -- $DEFAULT_JVM_OPTS $JAVA_OPTS $GRADLE_OPTS
"\"-Dorg.gradle.appname=$APP_BASE_NAME\"" -classpath "\"$CLASSPATH\""
org.gradle.wrapper.GradleWrapperMain "$APP_ARGS"
# by default we should be in the correct project dir, but when run from Finder on
Mac, the cwd is wrong
if [ "$(uname)" = "Darwin" ] && [ "$HOME" = "$PWD" ]; then
```

```
  cd "$(dirname "$0")"
fi
exec "$JAVACMD" "$@"
```