Show all of your work clearly in the space provided or on the additional page at the end of the exam. If the additional page is used, clearly identify to which exam question it is related. Be sure to **read each problem carefully**. Note that the exam is double sided.

1. (10 points) Precisely and concisely describe the **lock-modify-unlock** approach to version control.

2. (10 points) Precisely and concisely describe how the **copy-modify-merge** approach to version control differs from the **lock-modify-unlock** approach.

3. (5 points) Which approach to version control does subversion employ?

4. Suppose that your partner commits changes to a file in your repository after you have checked the file out. Suppose further that you have made modifications to the same file.
(a) (5 points) Explain why your attempt to commit the file to the repository will fail.

(b) (10 points) List all of the steps required using TortoiseSVN to successfully incoporate your changes to the file into the repository.

5. (10 points) Explain, in detail, what happens when following code is executed:

```
Thread thread = new Thread(new WorkerThread(), "Worker Thread");
thread.start();
```

6. (5 points) Explain what the following line of code does:

```
Thread.sleep(1000);
```

7. (15 points) Consider the following code (continues onto the next page).

```
public abstract class DesktopItem {

  private String name;
  private String extension;
  private String fullPath;

  public DesktopItem(String path, String name) { /* removed */ }

  public void rename(String name) { /* removed */ }

  public abstract void open();

  public void move(String path) { /* removed */ }

  public String getFullName() { /* removed */ return null; }

  public abstract int size();
}

public class Folder extends DesktopItem {

  java.util.List<DesktopItem> items;

  public Folder(String path, String name) { /* removed */ }

  public void open() { /* removed */ }

  public int size() { return 0; }
}
```

```
public class TextFile extends DesktopItem {

  private String text;

  public TextFile(String path, String name) { /* removed */ }

  public void open() { /* removed */ }

  public int size() { /* removed */ return 0; }
}
```

Draw the UML class diagrams for each of the following classes (assume each class is in a separate file). Be sure to show the datatypes for all attributes, methods, and method parameters and the relationship between the classes.

8. (10 points) The Exam1 class has the usual main() method and method called park(). Use the sequence diagram on page 7 to complete the Java program in the space below.

```
public class Exam1 {

  public static void main(String[] args) {









  }

  public static void park(Car car) {









  }
}
```

9. Use the ANT build file on page 8 to answer the following questions. Assume that ANT is installed and configured and that the build.xml file is located in the following directory: D:/SE2030.

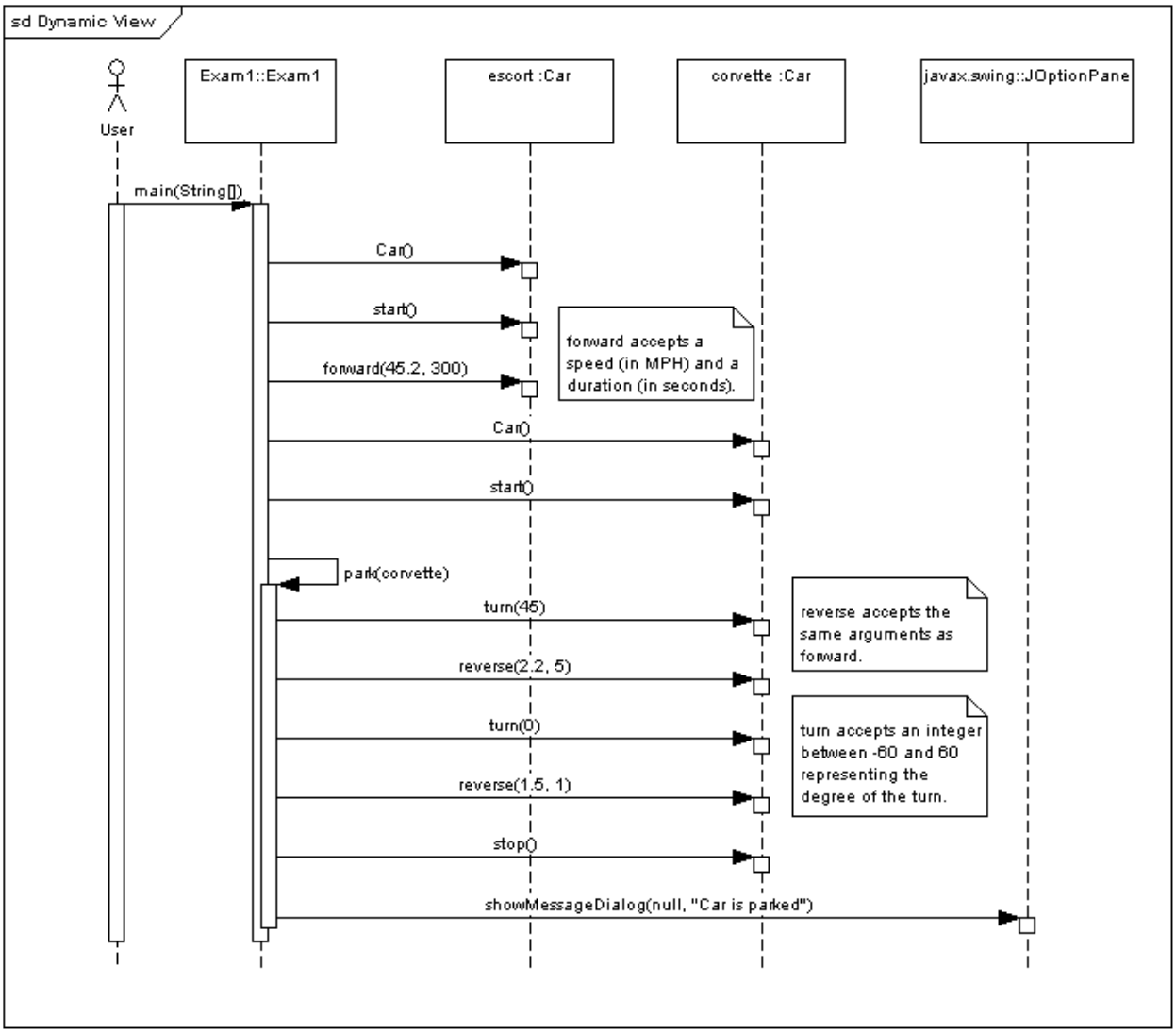
(a) (5 points) What will happen if the following command is entered at the command line?

```
D:\SE2030>ant clean
```

(b) (5 points) Where will ANT store the .class files generated by the compilation process? Give the absolute (full) path.

(c) (5 points) Where must the Driver.java file be located? Give the absolute (full) path.

(d) (5 points) What addition to the build.xml is necessary in order to tell the JVM that it should look in the winplotter.jar when executing the .jar created by the ANT script? Write your answer here (not on page 8).


sd Dynamic View
User
Exam1::Exam1
escort :Car
corvette :Car
javax.swing::JOptionPane
main(String[])
Car()
start()
forward(45.2, 300)
forward accepts a speed (in MPH) and a duration (in seconds).
Car()
start()
park(corvette)
turn(45)
reverse accepts the same arguments as forward.
reverse(2.2, 5)
turn(0)
turn accepts an integer between -60 and 60 representing the degree of the turn.
reverse(1.5, 1)
stop()
showMessageDialog(null, "Car is parked")

```
<?xml version="1.0"?>
<project name="Midterm" default="all" basedir=".">

  <property name="src_dir" value="./src"/>
  <property name="lib_dir" value="C:/data/db/se1010/jars"/>
  <property name="lib_jars" value="winplotter.jar"/>
  <property name="output_dir" value="./antbuild"/>
  <property name="class_dir" value="${output_dir}/classes"/>
  <property name="main_class" value="edu.msoe.se2030.taylor.Driver"/>

  <path id="classpath">
    <fileset dir="${lib_dir}" includes="**/*.jar"/>
  </path>

  <target name="prepare">
    <mkdir dir="${output_dir}"/>
    <mkdir dir="${class_dir}"/>
  </target>

  <target name="clean">
    <delete dir="${output_dir}"/>
  </target>

  <target name="compile" depends="prepare">
    <javac srcdir="${src_dir}" destdir="${class_dir}">
      <classpath refid="classpath"/>
    </javac>
  </target>

  <target name="deploy" depends="compile">
    <copy todir="${output_dir}" preservelastmodified="true">
      <fileset dir="${lib_dir}">
        <include name="**/*.jar"/>
      </fileset>
    </copy>
  </target>

  <target name="jar" depends="deploy">
    <jar jarfile="${ant.project.name}.jar" basedir="${class_dir}">
      <manifest>
        <attribute name="Main-Class" value="${main_class}"/>


      </manifest>
    </jar>
  </target>

  <target name="all" depends="clean,jar"/>

</project>
```