General instructions:

- There are **4** problems (150 points) on 7 pages.
- You are allowed to copy files to your exam machine during the first 15 minutes of the examination period.
- No files apart from ones stored in the exam machine can be accessed at any time during the examination period.
- Hard-copied or printed documents are allowed.
- Internet access is not allowed at any time during the examination period.
- Use **c:\temp** as Eclipse workspace for all projects in the exam.

(30 points) Problem 1: What were the times when…?

1. **Java project creation**:
   - Failure to finish this part will prohibit further grading of this problem.

   a. Set up a new Java project in Eclipse named "problem1" so that the location of the project is at **c:\temp\[STUDENT_ID]\ problem1** (We will only grade this problem by opening the project from this folder.)

   b. No source files are provided in this problem.

2. **Instructions**:

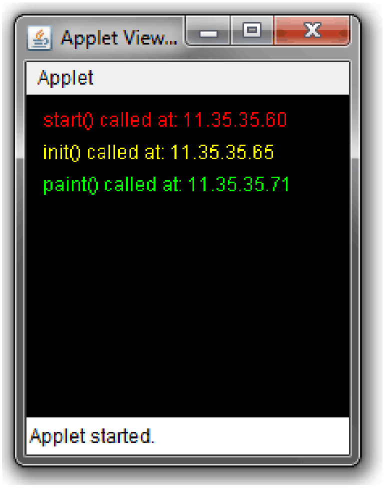
   a. Write a Java applet that shows the times when the most recent calling to *init()*, the most recent calling to *start()*, and the most recent calling to *paint()* were made.

      i. Apart from the applet itself, neither additional *Component* nor its subclasses (direct/indirect) can be created. This means that the showing of the time needs to be done on the *Graphics* object associated with the *Applet* object.

      ii. The message related to *start()* is shown in red, the one related to *init()* is shown in yellow, and the one related to *paint()* is shown in green. The messages are shown on a black background.

      iii. The times are shown in the following format: HOUR:MINUTE:SECOND:MILLISECOND

The program should look similar to the one shown below.



```
start() called at: 11.35.35.60
init() called at: 11.35.35.65
paint() called at: 11.35.35.71
```

```
Applet started.
```

Problem 1: Grading Checklist

Note that this is NEITHER a set of grading criteria NOR an answer sheet. Don't write anything on this page.

The project folder has your student ID.	
The project can be opened properly.	
The program can be compiled and executed.	
The latest times the three methods were called were obtained and shown correctly.	
HOUR, MINUTE, SECOND, MILLISECOND values were obtained and shown correctly.	
The colors were correct.	
No other *Components* were created.	

(30 points) Problem 2: Thread-powered Slot Machine

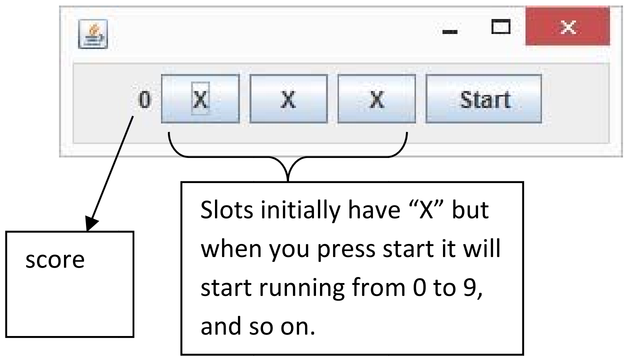
1. **Java project creation**:
   - Failure to finish this part will prohibit further grading of this problem.

   a. Set up a new Java project in Eclipse named "problem2" so that the location of the project is at **c:\temp\[STUDENT_ID]\ problem2** (We will only grade this problem by opening the project from this folder.)

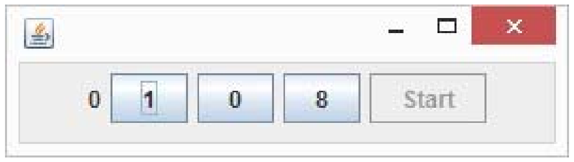
   b. You are provided with a jar file (**threadExam_Student.jar**) containing a simple slot machine program with source code.

      i. **threadExam_Student.jar** can be opened with WinRAR in order to obtained the source code.

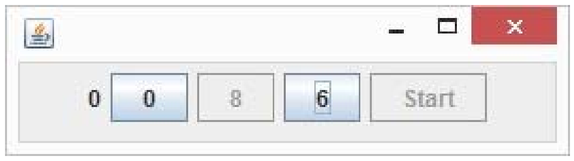
2. **Program Description**:
   A simple slot machine program is given as jar file. When run, it looks as follows:


0
X
X
X
Start
score
Slots initially have "X" but when you press start it will start running from 0 to 9, and so on.

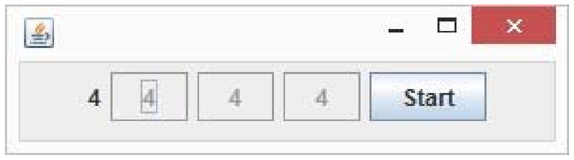
When running, the program repeatedly and randomly chooses a slot to increment a number within. Note that the start button becomes in active.



We can then click on a slot to stop it from moving.



Once we click all slots, a score will be updated (we only score if all stopped numbers are the same). The additional score is equal to the number in each slot. The start button becomes active again and we can continue from there.



3. **Rewrite the program**:

   Rewrite this program so that each slot has a thread associated with it and runs separately (all slots will now rotate at the same time, each with different speed). You must also update the user interface via an event dispatcher thread.

Problem 2: Grading Checklist

Note that this is NEITHER a set of grading criteria NOR an answer sheet. Don't write anything on this page.

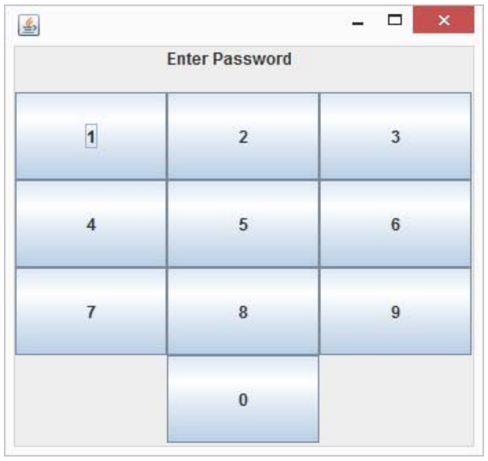
The project folder has your student ID.	
The project can be opened properly.	
The program can be compiled and executed.	
Each slot ran using its own Thread.	
GUI got updated through the event dispatcher thread.	

(60 points) Problem 3: Password Numeric Keypad

1. **Java project creation**:
   - Failure to finish this part will prohibit further grading of this problem.

   a. Set up a new Java project in Eclipse named "problem3" so that the location of the project is at **c:\temp\[STUDENT_ID]\ problem3** (We will only grade this problem by opening the project from this folder.)
   b. No source files are provided in this problem.

2. **Program Description**:
   A sample password check program looks like:

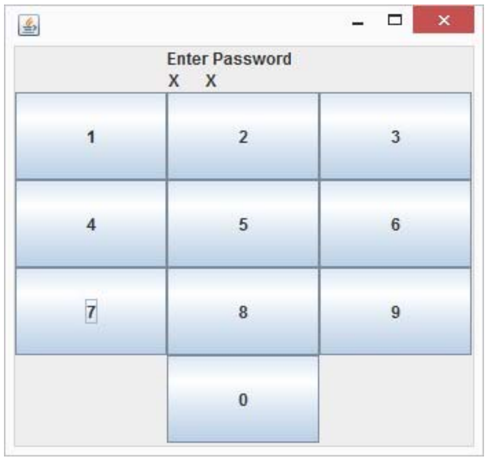


With the following variables for checking the hard-coded password (the password is 1234):

```
// for password checking
String[] password = { "1", "2", "3", "4" };
String[] enterPw = new String[4];

//index of the current position in enterPw
int usedIndex = 0;
```

When a button is pressed, the program remembers each one, and updates the user interface with "X" for each one letter of the password:



When all 4 letters are entered, if the password does not match, the program resets and the user has to enter the password again. Otherwise, the program shuts down and closes its windows without any error.

3. **Write the code**:
   - Create the program.
   - Set up the user interface for this program so that it looks like the sample page above.
   - Write code so that the program functions as specified above.

Problem 3: Grading Checklist

Note that this is NEITHER a set of grading criteria NOR an answer sheet. Don't write anything on this page.

The project folder has your student ID.	
The project can be opened properly.	
The program can be compiled and executed.	
The GUI widgets were laid out as in the given screenshot.	
The program behaved correctly.	

(30 points) Problem 4: Icon Maker

1. **Java project creation**:
   - Failure to finish this part will prohibit further grading of this problem.

   a. Set up a new Java project in Eclipse named "problem4" so that the location of the project is at **c:\temp\[STUDENT_ID]\ problem4** (We will only grade this problem by opening the project from this folder.)

   b. The source files provided for this problem are **IconMaker.java**, **IconPanel.java**, **PalettePanel.java** and **IconFileUtil.java**

2. **Program description**

   This program is a *JFrame* application that lets the user create 8 cells X 8 cells icon. In the program, the icon is represented as a sequence of 64 instances of *PixelPatch* class laid out by a *GridLayout* of a container. The user can save the current icon into a file which can be loaded back to the program at a later time. The loaded icon has exactly the same colors as when it was saved.

3. **Complete the program**

   a. Make modification to the program so that the "save" functionality works properly.
      i. You are NOT allowed to modify code segments related to the "load" functionality.

Problem 4: Grading Checklist

Note that this is NEITHER a set of grading criteria NOR an answer sheet. Don't write anything on this page.

The project folder has your student ID.	
The project can be opened properly.	
The program can be compiled and executed.	
The "save" functionality works properly.	
A saved icon can be loaded back properly.	
All other functionalities work properly (similarly to the one before the code was modified).	