



### Task 1: Model [30]

Zweck: Die Bilder der Puzzleteile sowie deren Position werden als Daten gespeichert.

- a) Erstellen Sie eine Klasse namens "Model" mit einer ganzzahligen Konstanten "SIZE", die auf den Wert 16 initialisiert ist! [2]
- b) Fügen Sie der Klasse zwei Felder (Arrays) als Instanzattribute hinzu: "images" vom Typ "Icon" und "position" vom Typ "int"! [2]
- c) Erstellen Sie eine Methode "sort" mit Zählschleife von 0 bis SIZE! [2]
- d) Weisen Sie im Schleifenrumpf dem jeweils aktuellen Element des "position"-Feldes den Index selbst zu, so dass Index des Feldes (Arrays) und enthaltener Wert identisch sind! [2]
- e) Erstellen Sie eine Methode "randomise", in welcher sequentiell (nacheinander) zwei Zählschleifen nach obigem Prinzip laufen! Weisen Sie in der ersten Schleife jedem Element des Feldes (Arrays) den ganzzahligen Wert -1 zu! [2]

Hinweis: Dies ist wichtig, um spätere Fehler beim Elementvergleich zu vermeiden!

- f) Rufen Sie innerhalb der zweiten Schleife eine Methode namens "getRandomPosition" auf! Weisen Sie ihren Ergebniswert dem aktuellen Element des Feldes (Arrays) zu! [2]
- g) Erstellen Sie die Methode "getRandomPosition" mit "int" als Rückgabety! Erzeugen Sie darin ein Objekt vom Typ "Random"! [2]
- h) Bauen Sie eine "while"-Endlosschleife ein! Erzeugen Sie darin mittels des "Random"-Objektes eine Zufallszahl zwischen 0 und (inklusive) 15! [2]
- i) Rufen Sie hiernach eine Methode namens "exists" auf, welcher die Zufallszahl als Argument zu übergeben ist! Nutzen Sie ihren Rückgabewert als Abbruchkriterium! [2]
- j) Implementieren Sie die Methode "exists"! Prüfen Sie darin mittels Zählschleife, ob das übergebene Argument im Feld (Array) bereits enthalten ist! Geben Sie in diesem Falle den Bool'schen Wert "true" zurück und beenden Sie die Schleife! [2]
- k) Erstellen Sie eine Methode namens "initialise"! Erzeugen Sie darin die beiden Instanzattribute mit identischer Größe! [2]
- l) Erzeugen Sie innerhalb einer Zählschleife jeweils ein neues Bild und speichern Sie es als Element im "images"-Feld (Array)! [2]
- m) Nutzen Sie den Schleifenlaufindex als Dateinamen! Fügen Sie "png" als Suffix sowie eventuell ein Paketverzeichnis als Präfix hinzu, damit die relative Pfadangabe korrekt ist! [2]

Hinweis: Die zu ladenden Bilddateien sind gegeben.

- n) Rufen Sie abschließend die "randomise"-Methode auf! [2]
- o) Erzeugen Sie das "Model"-Objekt in der "main"-Methode der Startklasse "Launcher"! [2]

Ergebnis: Alle nötigen Daten sind vorhanden und können nun dargestellt werden.

### Task 2: View Panel [20]

Zweck: Das Hauptfenster wird zusammengebaut.

- a) Erstellen Sie auf Basis eines Swing-Rahmenfensters eine Klasse namens "View"! [2]
- b) Geben Sie ihr die Attribute "model" vom Typ "Model" und "labels" vom Typ "JLabel[]"! [2]
- c) Erstellen Sie eine "initialise"-Methode, in welcher das Instanzattribut "labels" mit der Größe "Model.SIZE" erzeugt wird! [2]
- d) Geben Sie dem Rahmenfenster einen Titel! Optimieren Sie die Größe so, dass alle enthaltenen Elemente optimal angezeigt werden! [2]



- e) Weisen Sie eine Schließoperation zu, die die Anwendung beendet, sobald das Fenster geschlossen wird! Schalten Sie das Fenster sichtbar! [2]
- f) Erzeugen Sie (noch zuvor) ein neues "JPanel"-Objekt mit Gitternetzanordnung (4 x 4)! [2]
- g) Nutzen Sie eine Zählschleife, um in jeder Gitternetzzeile ein eigenes "JLabel"-Objekt zu platzieren! Fügen Sie es außerdem zum Instanzattribut "labels" hinzu! [2]
- h) Weisen Sie dem jeweiligen "JLabel"-Objekt als Icon das passende Bild aus dem Model hinzu! Nutzen Sie als Bildindex das aktuelle Element des "position"-Feldes (Arrays)! [2]
- i) Fügen Sie das "JPanel"-Objekt dem Fenster hinzu! Platzieren Sie es mittels "BorderLayout.CENTER"! [2]
- j) Erzeugen und initialisieren Sie ein "View"-Objekt in der "main"-Methode! Übergeben Sie ihm das "Model"-Objekt! [2]

Ergebnis: Beim Start der Anwendung sollte ein Fenster mit Bildern zu sehen sein.

### **Task 3: Mouse Controller [20]**

Zweck: Bilder sollen verschoben werden können.

- a) Erstellen Sie eine Klasse namens "MouseController", die Mausereignisse abfängt! Geben Sie ihr ein Instanzattribut "label" vom Typ "JLabel"! [2]
- b) Implementieren Sie die "mousePressed"-Methode! Vergessen Sie nicht den entsprechenden Aufruf der Methode der Superklasse! [2]
- c) Speichern Sie die Quelle des Ereignisses im "label"-Instanzattribut, falls der linke Mausknopf gedrückt wurde! [2]
- d) Speichern Sie anderenfalls die Quelle des Ereignisses in einer lokalen "JLabel"-Variablen! [2]
- e) Bestimmen Sie das jeweils aktuelle Icon des "label"-Instanzattributes sowie der lokalen "JLabel"-Variablen! Speichern Sie beide in je einer lokalen Variablen! [2]
- f) Vertauschen Sie nun die Icons der beiden "JLabel"-Objekte, indem Sie ihnen das jeweils andere Icon zuweisen! [2]
- g) Führen Sie eine Prüfung des "label"-Instanzattributes auf Null-Zeiger durch, um einen Programmabsturz zu vermeiden! [2]
- h) Erstellen Sie eine Klasse namens "Controller", welche ein Instanzattribut vom Typ "MouseController" hält! Geben Sie ihr eine "initialise"-Methode, worin das "MouseController"-Objekt erzeugt wird! [2]
- i) Erzeugen und initialisieren Sie das "Controller"-Objekt in der "main"-Methode! Übergeben Sie es als Argument an die "initialise"-Methode der "View"! [2]
- j) Fügen Sie in der "initialise"-Methode der "View" die im "Controller"-Objekt enthaltene "mouseController"-Eigenschaft jedem in der Zählschleife erzeugten "JLabel"-Objekt als "MouseListener" hinzu! [2]

Ergebnis: Das mit links angeklickte Bild wird zum mit rechts angeklickten verschoben.

### **Task 4: View Toolbar [10]**

Zweck: Eine Werkzeugleiste soll Funktionalität bereit stellen.

- a) Erzeugen Sie in der "initialise"-Methode der "View" eine Werkzeugleiste (Toolbar)! [2]
- b) Geben Sie ihr drei Knöpfe mit je einer der Aufschriften: "Exit", "Sort", "Randomise"! [2]
- c) Weisen Sie den drei Knöpfen ein eindeutiges "ActionCommand" zu! Geben Sie den drei Knöpfen einen sinnvollen Hinweistext (Tooltip) Ihrer Wahl! [2]

d) Fügen Sie zwischen den drei Knöpfen je einen Trennabstand ein! Aktivieren Sie den "Rollover"-Effekt! [2]

e) Ordnen Sie die Werkzeugleiste innerhalb des Hauptfensters oben an! [2]

Ergebnis: Die Werkzeugleiste wird im Fenster angezeigt.

### **Task 5: Action Controller [20]**

Zweck: Die eigentliche Funktionalität der Knöpfe der Werkzeugleiste wird ergänzt.

a) Erstellen Sie eine Klasse namens "ActionController" für das Verarbeiten von "ActionEvents"! [2]

b) Halten Sie dort Referenzen auf das "Model" und die "View" in je einem Instanzattribut! [2]

c) Fangen Sie das "exit"-Kommando ab! Lassen Sie es die Anwendung beenden! [2]

d) Fangen Sie auch die Kommandos "sort" und "randomise" ab! Rufen Sie die jeweils passende Methode der "Model"-Klasse auf! Veranlassen Sie ein Neuzeichnen des Fensters! [2]

e) Geben Sie auf der Konsole eine Fehlermeldung aus, wenn das abgefangene Kommando unbekannt ist! [2]

f) Definieren Sie in der "ActionController"-Klasse Konstanten für die "ActionCommands"! Verwenden Sie sie überall (auch in der "View") an Stelle hart kodierter Literale! [2]

g) Ergänzen Sie die "Controller"-Klasse durch ein zweites Attribut des Typs "ActionController"! Erzeugen Sie es in der dortigen "initialise"-Methode! [2]

h) Erweitern Sie diese "initialise"-Methode, so dass "Model" und "View" als Argumente übergeben und dem "ActionController"-Objekt als Attributwerte zugewiesen werden! [2]

i) Weisen Sie in der "initialise"-Methode der "View"-Klasse den Knöpfen der Werkzeugleiste den "ActionController" als "ActionListener" zu! [2]

j) Überschreiben Sie die geerbte "paint"-Methode in der "View"-Klasse! Weisen Sie per Zählschleife dem jeweiligen "JLabel"-Objekt als Icon das passende Bild aus dem Model zu! Nutzen Sie als Bildindex das aktuelle Element des "position"-Feldes (Arrays)! [2]

Ergebnis: Die Knöpfe der Werkzeugleiste funktionieren.

**Viel Erfolg!**