Examination
User Interaction (UI)

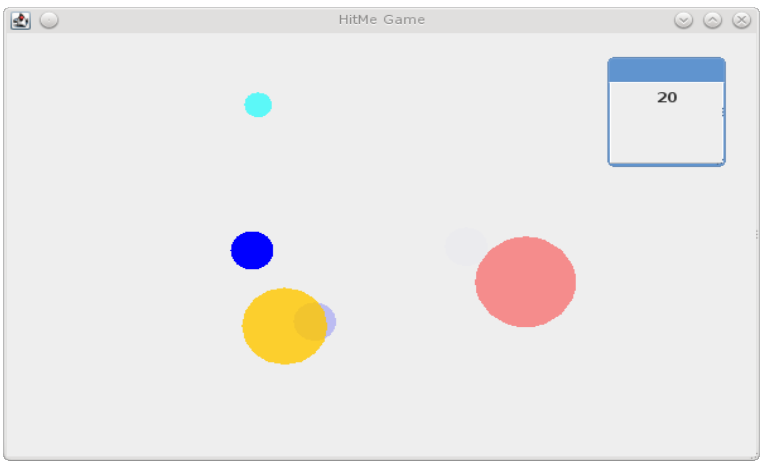
Prof. Dr.-Ing. Christian Heller <christian.heller@ba-leipzig.de>

Personal Data	
First and Last Name	Ihre Daten müssen von der Klausuraufsichtsperson schriftlich auf der Anwesenheitsliste festgehalten werden!
Matriculation Number	
Subject and Year	CS 2012
Login	Das Login "klaus..." muss unbedingt schriftlich auf der Anwesenheitsliste festgehalten werden, da sonst keine Zuordnung des Logins zu Ihrem Namen und damit keine Korrektur der Klausur möglich ist!

Examination Data	
Date	2013-12-12
Duration [min]	120
Permitted Study Aids	Dokumentation im lokalen Netzwerkverzeichnis (Intranet); NICHT gestattet sind Kommunikationsmöglichkeiten (Internet) oder Anmeldung via SSH auf dem Rechner "fileserv", wo Ihr Homeverzeichnis liegt, oder eine Anmeldung mit Ihrem Klausur-Login nach Ende der Prüfung. Dies kann leicht geprüft werden (last cs12*, Server-Log-Dateien). Bitte unterlassen Sie also Täuschungsversuche in Ihrem eigenen Interesse.
Remarks	Hinterlegen Sie alle Programme und Antworten in elektronischer Form! Es wird kein Papier angenommen. Möchten Sie Lösungen erläutern, so nutzen Sie Quelltext-Kommentare oder legen eine Text-Datei an. Speichern Sie sämtliche Daten im HOME-Verzeichnis des Nutzers, d.h. unter Windows auf Laufwerk H:\ (NICHT auf C:\ oder "Eigene Dateien")! Idealerweise legen Sie dort ein Unterverzeichnis namens "klausur" an. Lesen Sie die Aufgaben komplett durch, bevor Sie sie lösen! Die Reihenfolge der Lösung ist Ihnen überlassen. Probieren Sie immer, eine Aufgabe zu lösen, da auch auf richtige Teile nicht vollständiger Lösungen Punkte vergeben werden! Falls vom Prinzip her richtig, so werden auch alternative Lösungen akzeptiert. Sie dürfen beliebig viele Bildschirmausgaben von Werten in den Quelltext einbauen, um ein Programm besser nachvollziehen zu können. Diese Aufgabenstellung in Papierform können Sie nach dem Ende der Klausur behalten.

Evaluation								
Task	1	2	3	4	5	6	7	Summe
Desired Value [Credit Point]	10	15	15	30	15	10	5	100

Im Rahmen der Klausur soll das Spiel "Hit Me" erstellt werden, in welchem man durch Klicken auf Kreise verschiedener Größe Punkte sammeln kann. Zum Einsatz kommen soll dabei das Observable-Observer-Framework des Paketes "java.util". Hier ein Bildschirmschnappschuss:



Hinweis: Sichtbarkeiten sowie get-/set-Methoden werden in ALLEN Aufgaben der Einfachheit halber vernachlässigt und können nach Belieben gesetzt beziehungsweise ignoriert werden.

Task 1: Data Model with Standard Java [10]

In dieser Aufgabe wird das Datenmodell angelegt.

a) Erstellen Sie ein Projekt beliebigen Namens, welches ein Paket "hitme" enthält! Kopieren Sie die gegebene Datei mit der Klasse "Target" hinein! [2]

b) Erstellen Sie im Paket eine Klasse namens "Model" mit folgenden Attributen [2]:

- "targets" vom Typ ArrayList<Target>
- "score" vom Typ int

c) Erstellen Sie eine Methode "initialise", in welcher das Attribut "targets" erzeugt und das Attribut "score" auf den Anfangszahlenwert Null gesetzt wird! [2]

d) Lassen Sie die Klasse "Model" von "Observable" erben! Erstellen Sie eine Methode namens "markChanged", worin die geerbte Methode "setChanged" aufgerufen wird! [2]

e) Erstellen Sie schließlich eine Klasse "HitMe" mit "main"-Methode, in welcher das Model instanziiert und seine "initialise"-Methode aufgerufen wird! [2]

Task 2: Main View with Java Foundation Classes (JFC) [15]

In dieser Aufgabe wird das Hauptfenster implementiert.

a) Erstellen Sie eine Klasse namens "View", welche von "JPanel" erbt! [2]

b) Geben Sie ihr ein Attribut "model" vom Typ "Model", welches mit dem an die neu anzulegende "initialise"-Methode übergebenen Parameter initialisiert werden soll! [2]

c) Lassen Sie die Klasse von der "Observer"-Schnittstelle erben! Lösen Sie in der dadurch zu implementierenden Methode ein "repaint"-Ereignis aus! [2]

d) Überschreiben Sie die geerbte "paintComponent"-Methode, wobei darin zunächst jene der Super-Klasse aufzurufen ist! [2]

e) Instanziieren Sie die "View"-Klasse in "HitMe::main" und initialisieren Sie das Objekt! [2]

f) Erzeugen Sie dort außerdem ein Objekt des Types "JFrame" und speichern Sie es in einer lokalen Variablen! Fügen Sie ihm das vorher erzeugte "View"-Objekt als grafische Komponente hinzu (Hinweis: "add")! [2]

g) Weisen Sie dem Rahmenfenster eine Standard-Schließoperation zu, setzen seine Größe auf 600 x 400 Pixel und machen Sie es sichtbar! [3]

Hinweis: Bei probeweisem Start der Anwendung sollte nun ein Fenster zu sehen sein.

Task 3: Circle Drawing with Abstract Windowing Toolkit (AWT) [15]

In dieser Aufgabe werden Kreisobjekte gezeichnet.

a) Fahren Sie mit der Implementierung der "View::paintComponent"-Methode fort, indem Sie den als Parameter übergebenen Grafikkontext unter Verwendung seiner "create"-Methode kopieren, in den Typ "Graphics2D" umwandeln und in einer lokalen Variablen speichern! [2]

b) Definieren Sie ein Farbobjekt beliebigen Wertes und weisen Sie es dem Grafikkontext zu! [2]

c) Erstellen Sie unter Verwendung des Konstruktors "Ellipse2D.Double" ein Objekt beliebiger Größe, das einer lokalen Variablen des Types "Shape" zuzuweisen ist! Füllen Sie es mittels "fill"-Methode des Grafikkontextes mit der vorher definierten Farbe! [2]

Hinweis: Bei probeweisem Start der Anwendung sollte nun ein Kreis zu sehen sein.

d) Betten Sie das eben beschriebene Erstellen eines Kreises in eine "for-each"-Schleife ein, welche über alle "Target"-Objekte des "Model"-Attributes iteriert! [2]

e) Ersetzen Sie die vier (durch Sie vorher willkürlich vergebenen) Parameter des Konstruktors "Ellipse2D.Double" durch die Position und Größe des jeweiligen "Target"-Objektes! [2]

Hinweis: Für Breite wie Höhe kann des "Target"-Objektes "size"-Attribut verwendet werden. Für die x- und y-Position jedoch muss jeweils "size / 2" subtrahiert werden, da im "Target" die Koordinaten des Mittelpunktes gespeichert sind.

f) Bestimmen Sie die zu verwendende Farbe nunmehr nicht mehr durch Zuweisung eines absoluten Farbwertes, sondern durch Aufruf der "Target::fadeColour"-Methode! [3]

g) Geben Sie die Ressourcen des Grafikkontextes nach Verlassen der Schleife mittels "dispose"-Methode frei! [2]

Hinweis: Die Ergebnisse dieser Änderungen sind erst zu sehen, wenn ein Controller existiert.

Task 4: Controller for Timer Events [30]

Diese Aufgabe kümmert sich um das regelmäßige Zeichnen von Kreisen.

a) Erstellen Sie eine Klasse namens "Controller", deren zwei Attribute vom Typ "Model" und "View" via "initialise"-Methode zugewiesen werden! [4]

b) Lassen Sie die Klasse von der "ActionListener"-Schnittstelle erben und implementieren Sie geforderte Methoden! [2]

c) Fügen Sie der Klasse ein drittes Attribut des Types "javax.swing.Timer" hinzu! Erzeugen Sie das ihm zuzuweisende Objekt ebenfalls in der "initialise"-Methode, wobei seine Verzögerung auf 1 s eingestellt werden soll! Übergeben Sie das "Controller"-Objekt selbst als "ActionListener"! [4]

d) Starten Sie den Timer! [2]

e) Fügen Sie der Klasse ein viertes Attribut namens "box" des Types "Rectangle" hinzu! Erweitern Sie die "initialise"-Methode um einen Initialwert für das Attribut! [4]

f) Erzeugen Sie in der "Controller::actionPerformed"-Methode ein neues "Target"-Objekt und initialisieren Sie es durch Aufruf der entsprechenden Methode, wobei das "box"-Attribut als Parameter mitzugeben ist! [2]

g) Fügen Sie das neue "Target"-Objekt der entsprechenden Liste des Models hinzu! [2]

h) Markieren Sie das Model durch Aufruf der passenden Methode als "changed"! [2]

i) Informieren Sie alle das Model beobachtenden Objekte durch Aufruf der Methode "notifyObservers"! [2]

j) Erweitern Sie die Methode "HitMe::main", so dass ein "Controller"-Objekt erzeugt wird! Initialisieren Sie es durch Aufruf der dafür bereits implementierten Methode! [4]

Hinweis: Als dritter Parameter ist die Ausdehnung des Rahmenfensters mitzugeben, welche Sie mittels "getBounds"-Methode bestimmen können!

k) Fügen Sie außerdem das "View"-Objekt als "Observer" zum Model hinzu! [2]

Hinweis: Nun sollten aller Millisekunden neue farbige Kreise zu sehen sein.

Task 5: Controller for Target Removal [15]

Hier werden veraltete Kreise wieder entfernt.

a) Fügen Sie der Klasse "Model" ein Attribut "frequency" des Types "int" hinzu! Initialisieren Sie es in der "initialise"-Methode mit dem Wert 100! [2]

b) Fügen Sie der Klasse "Model" ein weiteres Attribut namens "time" des Types "long" hinzu! Initialisieren Sie es in der "initialise"-Methode mit dem Rückgabewert des Methodenaufrufes "System.currentTimeMillis()"! [2]

c) Erweitern Sie die Methode "Controller::actionPerformed" vor dem Abschnitt zum Erzeugen neuer "Target"-Objekte durch eine klassische "for"-Schleife! Verwenden Sie einen "Iterator" zum Durchlaufen der "Target"-Objekte, um Fehler beim Löschen zu vermeiden! [4]

d) Lassen Sie das "Target"-Objekt mittels "ageTarget"-Methode altern, wobei das Produkt aus "Model::time" und "Model::frequency" als Parameter mitzugeben ist! [2]

e) Entfernen Sie das "Target"-Objekt aus der Liste, wenn sein Alter "age" größer als seine Lebenszeit "lifespan" ist! [3]

Hinweis: Beim Testen sollte nun der vorherige Kreis verschwinden, bevor ein neuer erscheint.

f) Umrahmen Sie den gesamten Quelltext der "actionPerformed"-Methode durch folgenden [2]:

```
// The old time.
long o = this.model.time;
this.model.time = System.currentTimeMillis() / this.model.frequency;
if (this.model.time != o) {
    ... leave existing code here ...
}
```

Hinweis: Diese Neuberechnung des "Model::time"-Attributes auf Basis der Frequenz bewirkt, dass die Kreise schrittweise und nicht mehr sofort verschwinden.

Task 6: Controller for Event Handling [10]

In dieser Aufgabe sollen Mausereignisse verarbeitet werden.

a) Lassen Sie die Klasse "Controller" von einer geeigneten Superklasse erben, um Mausklicks abzufangen! Reagieren Sie durch Überschreiben einer passenden Methode auf sie! [2]

b) Filtern Sie Klicks mit der linken Maustaste heraus! Bestimmen Sie die Mauskoordinaten des Ereignisses und speichern Sie sie in einer lokalen Variablen! [2]

c) Iterieren Sie durch alle im Model gespeicherten "Target"-Objekte! Finden Sie mittels Methode der "Target"-Klasse heraus, ob der Mausklick innerhalb eines Kreises ("Targets") erfolgte! [2]

d) Erhöhen Sie, wenn dem so ist, die im Attribut "score" des Models gespeicherte Punktezahl in passender Weise, je nach "Target"! Entfernen Sie außerdem das "Target"-Objekt aus dem entsprechenden Container im Model! [2]

e) Markieren Sie desweiteren das Model durch Aufruf der passenden Methode als "changed" und informieren Sie alle das Model beobachtenden Objekte! [1]

f) Erweitern Sie die "View::initialise"-Methode so, dass das "Controller"-Objekt als zweiter Parameter übergeben und als Mauslauscher ("MouseListener") in der "View" registriert wird! [1]

Task 7: Score Board as Dialogue [5]

Ein Dialogfenster soll die aktuelle Punktzahl permanent anzeigen.

a) Erstellen Sie eine von "JPanel" erbende Klasse namens "ScoreBoard"! Instanziieren Sie sie in "HitMe::main"! Binden Sie sie (ähnlich der Klasse "View") in den "Notification"-Mechanismus ein, so dass die Punktzahl stets aktuell ist! [2]

b) Betten Sie das "ScoreBoard" in einen "JDialog" ein (z. B. in einer "initialise"-Methode)! [1]

c) Fügen Sie dem "ScoreBoard" ein "JLabel"-Objekt hinzu, welches durch Überschreiben der "paintComponent"-Methode auf den jeweils aktuellen Wert gesetzt wird! [2]

Viel Erfolg!