オブジェクト指向プログラミング H26 年度 後期中間試験 (2014.12.02 重村哲至)

IE5 ____ 番 氏名 模範解答

1. クラス図を参考に答えなさい。(3 点× 10 問= 30 点)

このアプリケーションを構成する主なクラスを MVC モデルに対応させると、PlayGroundPanel が **(1)** 、ToyBox クラスが **(2)** 、BallApp クラスが **(3)** の役割を持つと考えられる。

Toy クラスは x,y,c の 3 つの **(4)** と多数の操作を持っている。Toy クラスは抽象クラスであるのでインスタンスを生成 **(5)** 。Ball クラスは Toy クラスを **(6)** している。Ball クラスのインスタンスは生成 **(7)** 。

BallApp は ToyBox と PlayGroundPanel のインスタンスを、ToyBox は Toy のインスタンスを保持しする。このような関連を **(8)** と言う。特に BallApp と ToyBox や PlayGroundPanel のような関連は **(9)** と呼ばれる。

ToyBox は Toy との関連を 0 個以上持つことができる。これは”*” の **(10)** で表現される。

語群：(あ) オブジェクト、(い) コントロール、(う) ビュー、(え) モデル、(お) 依存、(か) 可視性、(き) 継承、(く) 合成集約、(け) 集約、(こ) 属性、(さ) 多重度、(し) 抽象操作、(す) 汎化、(せ) できる [(5,7) の候補]、(そ) できない [(5,7) の候補]

(1)	(う)	(2)	(え)	(3)	(い)	(4)	(こ)	(5)	(そ)	(6)	(き)	(7)	(せ)	(8)	(け)
(9)	(く)	(10)	(さ)												

2. ソースプログラム中、Toy クラス内の空欄 2 箇所には同じキーワードが入ります。何か答えなさい。(5 点)

```
abstract
```

3. ソースプログラム中、PlayGroundPanel の paintComponet メソッドの空欄に適切なプログラムを以下に書きなさい。(5 点× 2 問= 10 点)

空欄 (A)

```
super.paintComponent(g);
```

空欄 (B)

```
toy.draw(g);
```

オブジェクト指向プログラミング H26 年度 後期中間試験 (2014.12.02 重村哲至)

IE5 ____ 番 氏名 模範解答

4. クラス図、シーケンス図を参考に、ソースプログラム中 btnBall のアクションリスナーの actionPerformed メソッドの空欄 (C) に適切なプログラムを以下に書きなさい。なお、x,y,r はボールの座標 (x,y) と半径 (r) を表す。(10 点)

```
Ball b = new Ball(c, x, y, r);
toyBox.add(b);
playGround.repaint();
```

5. クラス図、シーケンス図を参考に、ソースプログラム中 btnBlock のアクションリスナーの actionPerformed メソッドの空欄 (D) に適切なプログラムを以下に書きなさい。なお、x,y,w,h は積み木の座標 (x,y) と大きさ (w:幅、h:高さ) を表す。(10 点)

```
Block b = new Block(c, x, y, w, h);
toyBox.add(b);
playGround.repaint();
```

6. クラス図、シーケンス図を参考に、ソースプログラム中 btnClear のアクションリスナーの actionPerformed メソッドの空欄 (E) に適切なプログラムを以下に書きなさい。(5 点)

```
toyBox.clear();
playGround.repaint();
```

オブジェクト指向プログラミング H26 年度 後期中間試験 (2014.12.02 重村哲至)

IE5 ＿＿＿ 番 氏名 模範解答

7. 全てのおもちゃを 100 ピクセル間隔で横一列に並べるボタン “Lineup” を追加します。このボタンを押すと、全てのおもちゃが (0,0),(100,0),(200,0),... の座標に一列に並びます。このボタンの actionPerformed メソッドを書きなさい。(15 点)

```
public void actionPerformed(ActionEvent e) {
  if (toyBox!=null) {

    for (int i=0; i<toyBox.size(); i++) {
      Toy toy = toyBox.get(i);
      toy.setX(i*100);
      toy.setY(0);
    }
    playGround.repaint();
  }
}
```

オブジェクト指向プログラミング H26 年度 後期中間試験 (2014.12.02 重村哲至)

IE5 ____ 番 氏名 模範解答

8. クラス図を参考に Block クラスを完成しなさい。なお、長方形の描画には Graphics クラスの fillRect(x,y,w,h) メソッドが使用できます。(15 点)

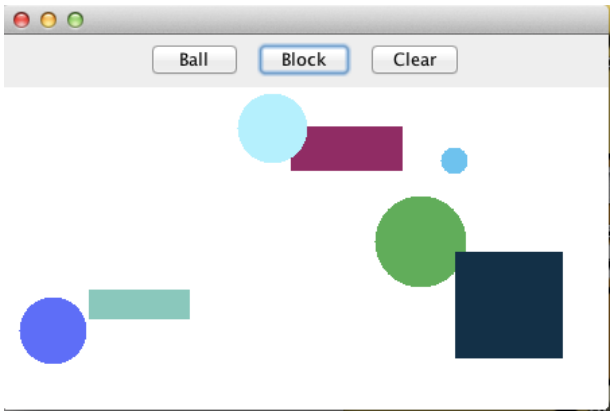
```
public class Block extends Toy {
 // 積み木の大きさ
 private int width;
 private int height;
 // コンストラクタ
 public Block(Color c, int x, int y, int w, int h) {
  super(c, x, y);
  this.width = w;
  this.height = h;
 }
 // アクセスメソッドの追加
 public int getWidth() {return width;}
 public void setWidth(int width) {this.width = width;}
 public int getHeight() {return height;}
 public void setHeight(int height) {this.height = height;}
 // 形状の描画
 @Override
 public void draw(Graphics g) {
  g.setColor(getColor());
  g.fillRect(getX(),getY(),width,height);
 }
}
```

オブジェクト指向プログラミング H26 年度 後期中間試験 (2014.12.02 重村哲至)

IE5 ____ 番 氏名 模範解答 (5/ 5)

はじめに

この試験では、図ような Java アプリケーションについて問題を出します。アプリケーションが表示しているのは、「おもちゃ」(ボールや積み木) が散らかった遊び場をイメージした表示です。ボタンを押す度に新しい「おもちゃ」が追加されます。



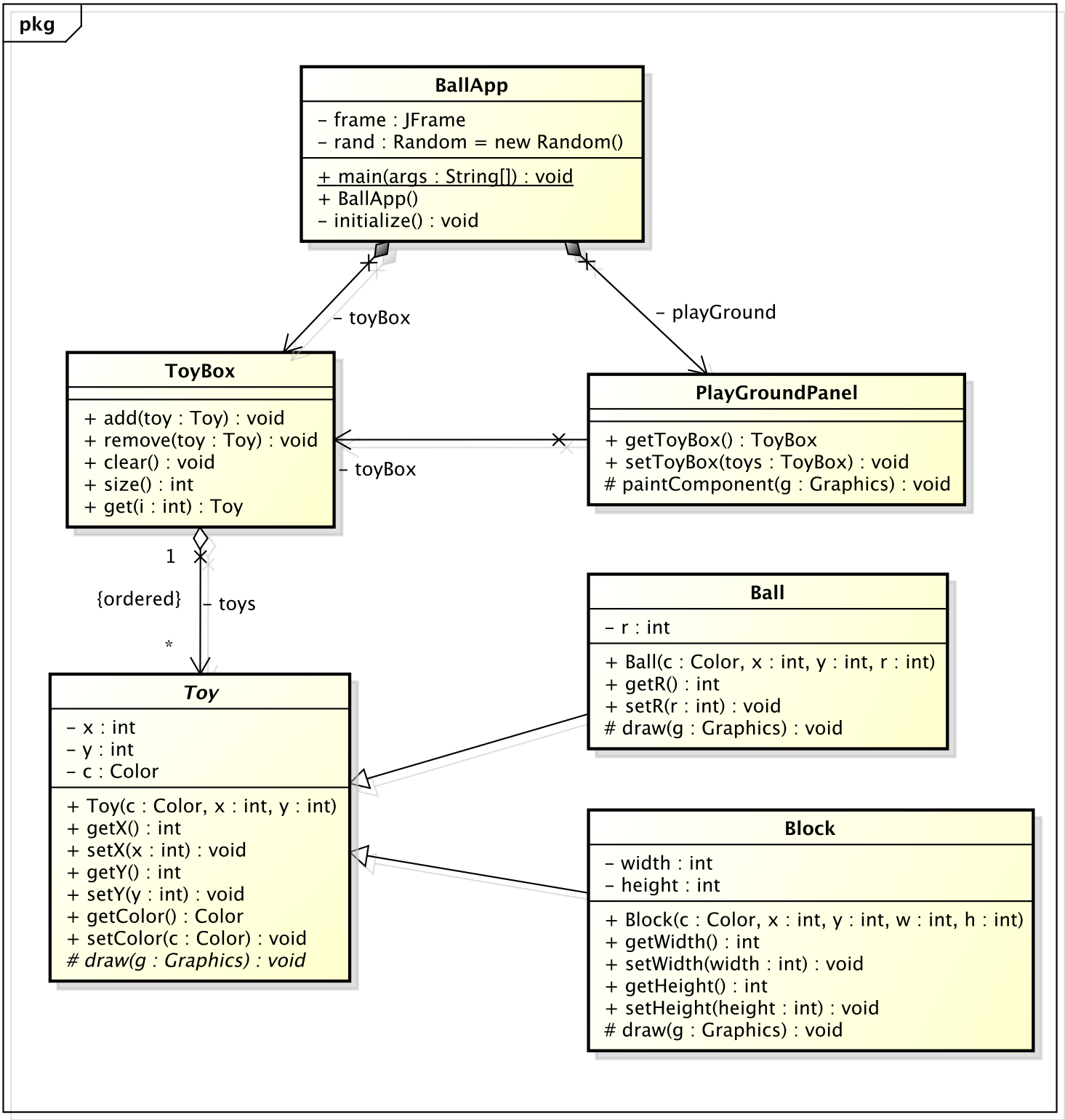
- “Ball” ボタンが押される度に画面上に新しいボール (円盤) を追加する。
- “Block” ボタンが押される度に画面上に新しい積み木 (長方形) を追加する。
- “Clear” ボタンは全てのボールと積み木を消去する。
- ボールと積み木の色と表示される座標は乱数で決まる。
- このアプリケーションのソースプログラムは [別紙 1] のとおりである。
- このアプリケーションのクラス図は [別紙 2] のとおりである。
- このアプリケーションのシーケンス図は [別紙 3] のとおりである。

[別紙１] ソースプログラム

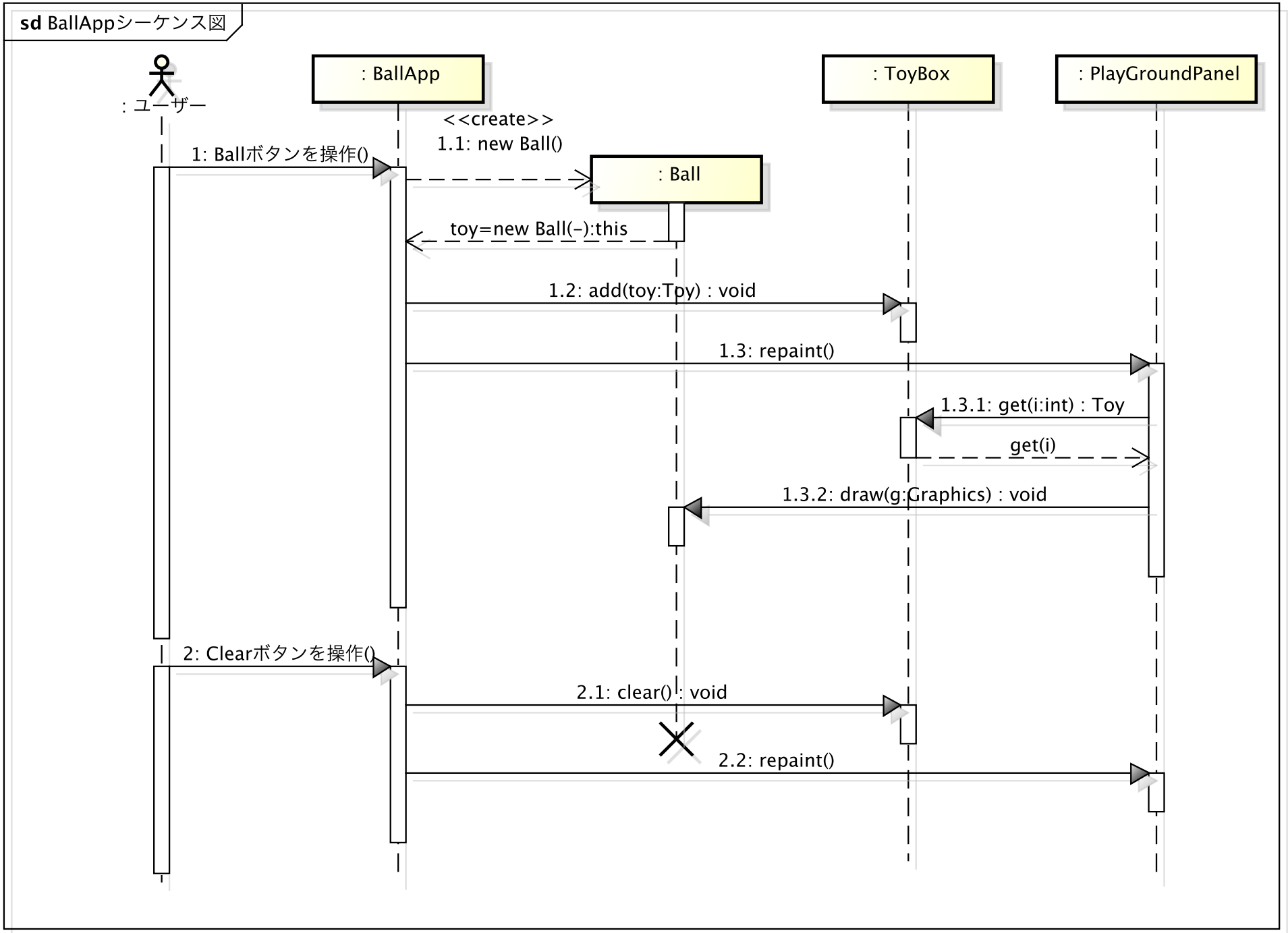
```
---------------------------------- (BallApp.java) -----------------------------
// アプリケーション本体
import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.BorderLayout;
import java.awt.Color;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.util.Random;

public class BallApp {
 private JFrame frame;
 // 乱数生成器
 private Random rand = new Random();
 // 遊び場とおもちゃ箱
 private PlayGroundPanel playGround;
 private ToyBox toyBox = new ToyBox();

 /**
  * Launch the application.
  */
 public static void main(String[] args) {
  EventQueue.invokeLater(new Runnable() {
   public void run() {
    try {
     BallApp window = new BallApp();
     window.frame.setVisible(true);
    } catch (Exception e) {
     e.printStackTrace();
    }
   }
  });
 }

 /**
  * Create the application.
  */
 public BallApp() {
  initialize();
  playGround.setToyBox(toyBox);
 }

 /**
  * Initialize the contents of the frame.
  */
 private void initialize() {
  frame = new JFrame();
  frame.setBounds(100, 100, 450, 300);
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  playGround = new PlayGroundPanel();
  playGround.setBackground(Color.WHITE);
  frame.getContentPane().add(playGround, BorderLayout.CENTER);

  JPanel controller = new JPanel();
  frame.getContentPane().add(controller, BorderLayout.NORTH);

  JButton btnBall = new JButton("Ball");
  btnBall.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {        // ボールの色(c)
    Color c = new Color(rand.nextInt(256),rand.nextInt(256),rand.nextInt(256));
    int r = rand.nextInt(40)+10;                       // ボールの半径(r)
```

```
    int x = rand.nextInt(playGround.getWidth()-2*r);   // ボールの座標(x,y)
    int y = rand.nextInt(playGround.getHeight()-2*r);
    /*** (C) ***/
   }
  });
  controller.add(btnBall);

  JButton btnBlock = new JButton("Block");
  btnBlock.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {        // 積み木の色(c)
    Color c = new Color(rand.nextInt(256),rand.nextInt(256),rand.nextInt(256));
    int w = rand.nextInt(90)+10;                       // 積み木の幅(w)
    int h = rand.nextInt(90)+10;                       // 積み木の高さ(h)
    int x = rand.nextInt(playGround.getWidth()-w);     // 積み木の座標(x,y)
    int y = rand.nextInt(playGround.getHeight()-h);
    /*** (D) ***/
   }
  });
  controller.add(btnBlock);

  JButton btnClear = new JButton("Clear");
  btnClear.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
    /*** (E) ***/
   }
  });
  controller.add(btnClear);
 }
}

---------------------------- (PlayGroundPanel.java) --------------------------
// おもちゃが散らばる遊び場パネル
import java.awt.Graphics;
import javax.swing.JPanel;

@SuppressWarnings("serial")
public class PlayGroundPanel extends JPanel {
 // おもちゃ箱
 private ToyBox toyBox;
 // アクセスメソッド
 public ToyBox getToyBox() {return toyBox;}
 public void setToyBox(ToyBox toys) {
  this.toyBox = toys;
  repaint();
 }
 // おもちゃ箱中の全部のおもちゃを描画する
 @Override
 protected void paintComponent(Graphics g) {
  /**** 空欄(A) ****/
  if (toyBox!=null) {
   for (int i=0; i<toyBox.size(); i++) {
    Toy toy = toyBox.get(i);  // おもちゃを一つ取り出し
    /*** 空欄(B) ***/         // 描画する
   }
  }
 }
}

-------------------------------- (ToyBox.java) ---------------------------------
// おもちゃ箱
import java.util.Vector;

public class ToyBox {
 private Vector<Toy> toys = new Vector<Toy>();   // Toy クラスのインスタンスを
                                                 // 格納する可変長配列クラス
```

```
 public void add(Toy toy) {toys.add(toy);}        // 配列の最後に追加する
 public void remove(Toy toy) {toys.remove(toy);} // 配列から削除する
 public void clear() {toys.clear();}              // 配列を空にする
 public int size() {return toys.size();}          // 配列の要素数を調べる
 public Toy get(int i) {return toys.get(i);}      // 配列の i 番目の要素を返す
}

---------------------------------- (Toy.java) ---------------------------------
// おもちゃ全般を表現する抽象クラス
import java.awt.Color;
import java.awt.Graphics;

public abstract class Toy {
 private int x;
 private int y;
 private Color c;
 // コンストラクタ
 public Toy(Color c, int x, int y){
  this.c=c;this.x=x;this.y=y;
 }
 // アクセスメソッド
 public int getX() {return x;}
 public void setX(int x) {this.x = x;}
 public int getY() {return y;}
 public void setY(int y) {this.y = y;}
 public Color getColor() {return c;}
 public void setColor(Color c) {this.c = c;}
 // 形状の描画
 abstract protected void draw(Graphics g);
}            public

---------------------------------- (Ball.java) ---------------------------------
//  ボールを表現するクラス
import java.awt.Color;
import java.awt.Graphics;

public class Ball extends Toy {
 // ボールの大きさ
 private int r;
 // コンストラクタ
 public Ball(Color c, int x, int y, int r) {
  super(c, x, y);
  this.r=r;
 }
 // アクセスメソッドの追加
 public int getR() {return r;}
 public void setR(int r) {this.r = r;}
 // 形状の描画
 @Override
public protected void draw(Graphics g) {
  g.setColor(getColor());
  g.fillOval(getX(),getY(),r*2,r*2);
 }
}

---------------------------------- (Block.java) ---------------------------------
// 積み木を表現するクラス(Block.java)
import java.awt.Color;
import java.awt.Graphics;

public class Block
 /* 以下省略 */
```


pkg
BallApp
- frame : JFrame
- rand : Random = new Random()
+ main(args : String[]) : void
+ BallApp()
- initialize() : void
- toyBox
- playGround
ToyBox
+ add(toy : Toy) : void
+ remove(toy : Toy) : void
+ clear() : void
+ size() : int
+ get(i : int) : Toy
- toyBox
PlayGroundPanel
+ getToyBox() : ToyBox
+ setToyBox(toys : ToyBox) : void
# paintComponent(g : Graphics) : void
1
{ordered}
- toys
*
Ball
- r : int
+ Ball(c : Color, x : int, y : int, r : int)
+ getR() : int
+ setR(r : int) : void
# draw(g : Graphics) : void
Toy
- x : int
- y : int
- c : Color
+ Toy(c : Color, x : int, y : int)
+ getX() : int
+ setX(x : int) : void
+ getY() : int
+ setY(y : int) : void
+ getColor() : Color
+ setColor(c : Color) : void
# draw(g : Graphics) : void
Block
- width : int
- height : int
+ Block(c : Color, x : int, y : int, w : int, h : int)
+ getWidth() : int
+ setWidth(width : int) : void
+ getHeight() : int
+ setHeight(height : int) : void
# draw(g : Graphics) : void

別紙２：クラス図


sd BallAppシーケンス図
: ユーザー
: BallApp
: ToyBox
: PlayGroundPanel
<<create>>
1.1: new Ball()
: Ball
1: Ballボタンを操作()
toy=new Ball(-):this
1.2: add(toy:Tpy) : void
1.3: repaint()
1.3.1: get(i:int) : Toy
get(i)
1.3.2: draw(g:Graphics) : void
2: Clearボタンを操作()
2.1: clear() : void
2.2: repaint()